Time-Optimal Control by Iterating Forward and Backward in Time

Adam Bäckström

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5944--SE
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2014

Abstract

When costumers look for a robot for their factory two things are important. The robot should be as efficient as possible, but still be cheap. In order to make the robot efficient the robot-controller has to know the dynamics of the robot and its limits. Based on these it can then generate a time-optimal plan (trajectory) for each movement. The standard way of generating a time-optimal trajectory with the exact dynamics and limits is very computationally heavy. Today most approaches do the planning based on the hardest limits on each axis of the robot. These values are then used even though the current limits might allow the robot to move faster. This means that the full capacity of the robot will not be utilized and because of this the efficiency is lowered.

In this thesis a different method for generating time-optimal trajectories is tested. The approach is based on iterating forward and backward in time and finding the point where the two paths meet. This approach has the advantage of that it is based on simulating the system. Therefore more complex dynamics can be included in the planning by just calculating a value instead of complicating the optimization problem. Another benefit is that robot manufacturers usually create simulation models of new robots already. This means that very little extra effort would be needed to create the trajectory generator using this approach and this reduces development costs.

Four different approaches for patching together the forward and backward paths are discussed in the thesis. The different techniques are tested on a simplified model of one servo axis of a robot and compared against a known time optimal solution for the simplified model. One of the techniques shows very good results and generates trajectories that are time-optimal.

Acknowledgements

Hereby, I want to thank my supervisors Klas Nilsson, from computer science at Lund University, and Anders Robertsson, from the department of Automatic Control at Lund University, for giving me the opportunity to work with this new technique for time-optimal control and explaining the idea to me. I also want to thank them and my supervisor Peter Valdt from B & R Automation in Malmo for answering my questions and helping me with anything I needed along the way.

In addition I want to thank Silvan Meile, exchange student from ETH, for lots of helpful discussions during my work and also for helping me to setup the Reflexxes library quickly for the comparison part of the thesis.

Contents

1

Introduction

A typical task for a robot consists of moving between different points. How it should do this is determined by a trajectory, which is a list of intermediate points to go to at given times. In order to utilise the robot's full potential one wants this trajectory to be as fast as possible. To do this a time-optimal problem has to be solved. The general way to do any kind of optimization is to start with a performance index [5]

$$L(u) \tag{1.1}$$

which is a function chosen such that its value increases the further away from the desired behaviour the parameter vector u yields. In optimal control this u can be seen as the control signal to the system. By minimizing the performance index over u the best solution u^0 is obtained. In the simplest case, without any constraints on u and when $L(u)$ has first and second partial derivatives everywhere, the minimum can be found based on the necessary conditions for a minimum [5]

$$\frac{\delta L}{\delta u} = 0, \frac{\delta^2 L}{\delta u^2} \geq 0 \tag{1.2}$$

If there are constraints on u or on the states of the dynamical system then the problem becomes more complicated, but the general idea is the same.

Let's say that we can described the dynamics of our robot by the following regression

$$x_{k+1} = f(x_k, u_k) \tag{1.3}$$

This will typically be a highly non-linear function because the moment of inertia for each joint usually depends non-linearly on the angles of all the other joints. We now want to optimize this system at a number of sample points and this is done by introducing the performance index

$$J = \Phi(x_N) + \sum_{i=0}^{N-1} L^i(x_i, u_i) \tag{1.4}$$

where $\Phi(x_N)$ and L^i are “the terminal cost" and a possibly different cost or performance index for each sample, respectively. This optimization problem can be handled in a way similar to Eq. (1.1) but now with N equality constraints from the regression equation Eq. (1.3). Typically one also adds inequality constraints on the control signal and states of the system which also has to be considered at each time step, further increasing the complexity of the optimization.

In the case of time-optimal control the following cost functions are used [5]

$$\Phi(x_N) = 0,\ L^i(x_i, u_i) = 1 \tag{1.5}$$

such that Eq. (1.4) becomes

$$J = N - 1 \tag{1.6}$$

which means that the optimized solution will minimize the number of time-steps between the given states at the start and end-point, thus minimizing the travel time. This approach will yield the true time-optimal trajectory that takes advantage of the non-linear dynamics of the robot. For example it might be better to initially let one joints move away from the end-point in order to lower the moment of inertia for the rest of the joints. However, in order to generate this time-optimal trajectory a two-point boundary-value problem has to be solved, which can be rather difficult even with a high speed computer [5]. This approach is therefore not suited for online trajectory generation and this gets more and more important as we want to make smarter robots that can react fast on sensor data.

Today most approaches for trajectory generation in robotics are based on worst case scenarios. This means that the non-linear dynamics in Eq. (1.3) is linearized by using the maximum moment of inertia for each joint as a constant. By using a linear model of the system the optimization problem becomes linear and the solution becomes a so called "bang-bang" control [5], which in many cases can be solved analytically. The drawback with this approach is that the generated trajectory will limit the robot unnecessarily hard in most cases and therefore it becomes less efficient. Say for example that we have a robot arm. Each axes of this arm will have torque limits. If the robot should go from a position where it has a stretched out pose to a position where it is folded together, then the inertia for the inner most axes would decrease during the movement. This means that the maximum acceleration would be low in the beginning and increase at the end. If the worst case inertia would be used for planning the movement then this low maximum acceleration would be used throughout the trajectory. For most parts, the acceleration would therefore be lower than the actual limit of the robot. The trajectory would because of this, for example, start braking earlier than necessary. This in turn leads to that the velocity would be lower than necessary at the end. Because of this, and other effects, it would therefore take the robot longer time to reach the end-point.

The initial idea behind this thesis is to linearize the system using the best case scenario instead of the worst case. The resulting trajectory would instead overesti-

mate the robots capacity and in most cases not be possible to follow (not be feasible). By inserting extra time between the points that are not feasible, giving the robot more time to catch up, it would be possible to transform the trajectory to something feasible. The idea with inserting extra time to make a trajectory feasible was tested by Ola Dahl in his PhD thesis [6]. He compensated for model errors, at run time, when following a time-optimal trajectory by sampling the trajectory at a different rate to insert extra time. This was done based on the current acceleration measurement, which has the problem that it is very noise sensitive. His approach aimed on compensating for small changes to the dynamics by doing small changes to the trajectory. Because they were small the importance of that the changes should be time-optimal was not that high. The main goal was to follow the desired path without deviating from it even if the actuator limitations were reached.

The idea here is instead to be able to handle large, but known, differences between the best case scenario and the non-linear model of the system. Since the changes may be large they need to be time-optimal in themselves in order to yield a time-optimal trajectory. In order to do this we basically need to solve a time-optimal problem going in between the non-feasible points in the trajectory, but we want to do this without minimizing a performance index. If the general approach of minimizing a performance index would be used the method would be too computationally heavy to serve any purpose.

The hypothesis for this thesis is that, instead of minimizing a performance index, a time-optimal solution can be generated by iterating forward and backward in time from the boundary conditions at $i = 0$ and $i = N$. Since the approach is based on iterating in time the dynamic moment of inertia (and other complex dynamics) can be added to the solution by simply calculated their values instead of increasing the complexity of the minimization problem.

Another benefit of using this time iterating approach is that it is based on simulating the system. Most robot manufacturers already have these models to be able to simulate and test the behaviour of the robot. This means that this approach easily can be used for new robots by using already existing models. It could therefore be a very time and cost effective method for creating a good trajectory planner for new robots.

Let's look closer at how the time-iterating approach would work. Say that we have one discrete-time model in forward time and one backward time model and that these models handle the limits on the states and the control signal. The backward time model is used to iterate backward as fast as possible from the states given at the finish ($i = N$) and the forward model is used to iterate forward from the states given at the start ($i = 0$) as fast as possible. If we can find a point at which all the states of the two cases are equal, then we can patch together the two paths. Since the two separate paths are time-optimal for leaving their initial states the complete path will be optimal for going between the states given at the start and the finish.

1.1 Car Example

Consider a car on a straight road, with a given maximum force and a speed limit on the road. The fastest way to leave the starting position is to apply the max force, see Fig. 1.1. At some point the speed limit will be reached and the force has to be lowered to keep the constant maximum speed. All this is straight forward.

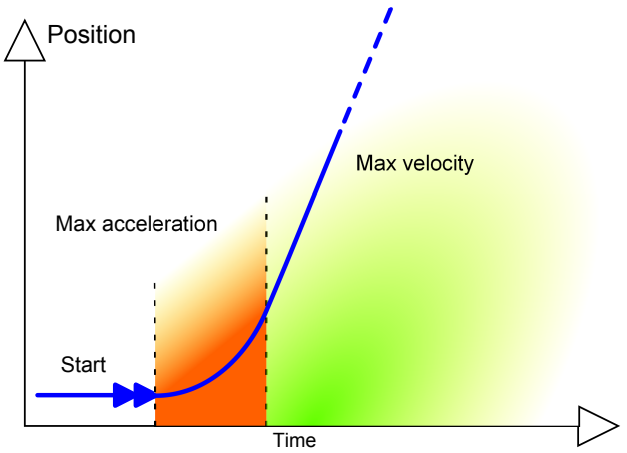

Figure 1.1 This figure shows the different stages of a car leaving the starting point in the quickest way.

Let's say that the car now is to stop at a certain position from going at the maximum speed. The quickest way to reach the stop position is to keep the maximum speed as long as possible and then apply the maximum braking force until the car has stopped. The problem is then to knowing when to start applying the maximum braking force in order to stop at the exact position. This is not straight forward and especially not if the jerk (force change) is restricted as well.

Say that the car already is standing at the stop and we can invert the direction of time. Going backwards the fastest way to leave the stop position is to apply the maximum braking force (fastest way to stop in forward time is fastest way to accelerate in backwards) and once the speed limit is reached increase the force to keep the speed constant, see Fig. 1.2. So in backwards time the system behaves very much like the acceleration case in forward time.

By using both forward and backward time one can find an intersection, see the red crosses in Fig. 1.3, between the two cases where both the speed and position are the same. At this point it is possible to switch from the accelerating case to the braking and reach the stop. Since both cases are time-optimal the total movement from start to stop will be time-optimal.

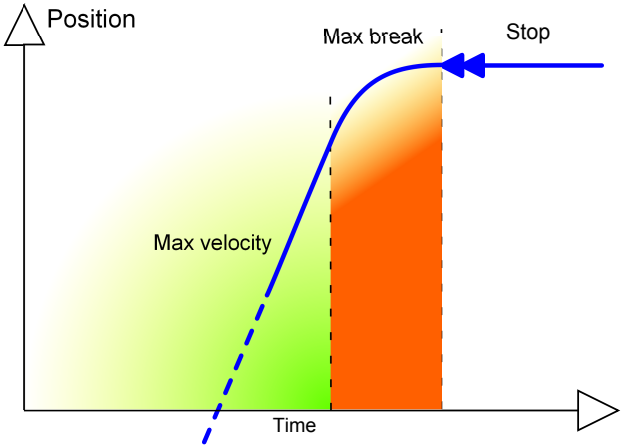

Figure 1.2 This figure shows the different stages of a car leaving the finish position, in the quickest way, in backwards time.

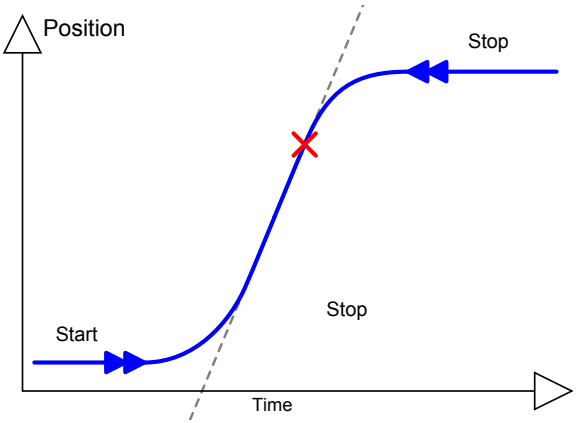

Figure 1.3 This figure shows two example intersections between forward and backward paths.

1.2 Robot example

A robot consists of a number of actuated joints, which are called axes. Each axis adds one degree of freedom to the robot. This means that getting the robot to go from one point to another typically is a multidimensional problem. The car in the example above only moves along one line (the road) and because of this there will always exist an intersection. For two or more dimensions there may not be an intersection. Therefore, it is not possible to take the robot dynamics and use it straight away.

Instead the dynamics has to be split up into the different parts, one for each axis. This can be viewed as having one car for each axis, each with different mass, engine strength and start and finish position.

One could solve the time-optimal problem for each axis separately, but this will not work because it will take each axis a different amount of time to reach its end-point. Say that the robot is not supposed to stop at the next point, but instead continue towards a new point. This means that the axes will not have zero velocity at the end-point. If one axis then reaches its end-point before the others it cannot stand and wait for the others to catch up. In order for the robot to actually reach the desired finish position all of the axes has to be at their individual finish positions at the same time. Therefore, the timing between the axes ultimately limits the resolution of the robot. This need for good synchronisation is also the reason why it is important that the trajectory does not exceed the capacity of the robot. If the trajectory does exceed the capacity then some of the axes will fall behind and result in poor synchronisation.

One way to split up the dynamics and still achieve synchronisation is to move all the axes as slow as the slowest one. This should be done by first iterating one time-step with the maximum capacity of each axis (as in the car example), which results in a state vector for each axis. Then compare how far each axis got, in terms of percentage of the total distance between start and finish. The lowest percentage of the axes is then used to calculate the corresponding position for each axis. This position is then used to calculate a state vector for each axis. By doing this, at each time-step, for both the forward and backward case it would be possible to find a point where all the states of all the axes are aligned. The solution that this yields will however not be the same as the one generated with the performance index discussed before. This comes from the fact that the method described here will force the robot to move along a straight line, between start and finish, in its joint space. Therefore, one axis cannot move in the wrong direction to lower the inertia for the other axes as the performance-index-approach allows.

1.3 The Robot

The robot used as a test basis for this thesis was an ABB IRB 340 Flexpicker [10], see Fig. 1.4. Four ACOPOS 1045 [1] drivers from B&R Automation and an X20 CP 1486 PLC [14] were used for controlling the robot. This particular set-up was put together by Rosenquist as his master thesis work [13]. The tests conducted on the robot during this thesis were all based on his original Automation studio project.

1.4 Outline

First a continuous-time model of one axis of the robot was created, see Sec. 3.2. The load of the axis was modelled with a constant inertia and constant external torque.

Figure 1.4 A picture of the ABB IRB 340 robot used in the experimental set-up.

This model was then used to create one forward and one backward discrete-time model. Because of problems with unstable dynamics of the backwards time model, see Sec. 2.7, a triple integrator model was used instead. During the work it became clear that only two states can be aligned using forward and backward iteration, see page 31. Because a triple integrator has three states a synchronisation stage between the forward and backward paths was added to handle the alignment of the third state. Due to the discrete time steps for the iterating it was not possible to get a perfect alignment of the states by just using the iterative models. Different approaches to handle the problems with discrete-time were tested, see Secs. 3.4-3.7. The results were compared against equivalent simulations for the Reflexxes library [11].

Apart from the time-optimal work a symbolic rigid-body dynamic model for the Flexpicker was created and translated to C-code to run on the PLC, see Sec. 2.3, but the model was not tested.

2

Theory

2.1 The Robot

ABB IRB 340 Flexpicker [10] is a parallel kinematic robot that has three identical arms which each consists of an upper arm, an elbow joint and a lower arm. The lower arm consists of two parallel links and two springs strapping them to the upper arm and travelling plate. The upper arms move in planes rotated 120 degrees to each other. Apart from the three elbow arms there is a fourth joint, the middle rotational axis, which adds one degree of freedom so that the robot can rotate horizontally. Figure 2.1 shows a sketch of the robot with the names of the different parts.

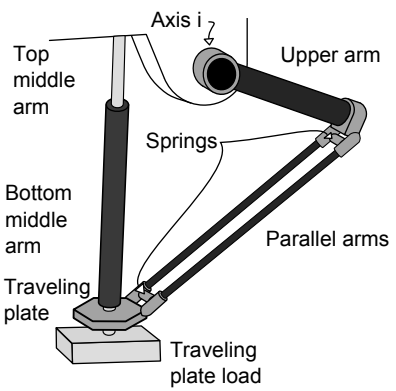

Figure 2.1 This figure shows the names of the different part of the robot with only one of the three arms drawn

2.2 Inverse Kinematics

In order to define movements in Cartesian coordinates the inverse kinematics of the robot has to be known, which is based on the geometry of the robot. The coordinate system is chosen to have its origin at the middle point between the three axes and the y-axis pointing up and the x-axis pointing towards axis1, see Fig. 2.2. The distance from the origin to an axis is called r_0 and the distance from an axis to its elbow joint is called r_1; β is the angle between the upper arm and the straight line between an axis and its elbow joint; r_2 is the length of one parallel arm and r_3 the distance from the centre of the travelling plate to the point where the lower arms attach. (p_x, p_y, p_z) is the desired Cartesian position.

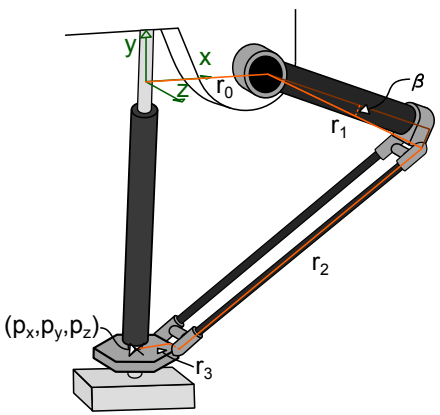

Figure 2.2 This figure shows how the coordinate system is defined and specifies the distances and angles used in the inverse kinematics. (p_x, p_y, p_z) is the wanted Cartesian position.

The first step in finding the angle for an axis is to project the geometry to its plane [13], see Fig. 2.3. For axis 1 this plane is the x-y plane which gives

$$l_1 = \sqrt{(r_0 - p_x - r_3)^2 + p_y^2}$$
$$l_2 = \sqrt{r_2^2 - p_z} \tag{2.1}$$

$$\cos(\phi_1) = \frac{p_x + r_3 - r_0}{l_1} \tag{2.2}$$

ϕ_2 is given by the cosine theorem

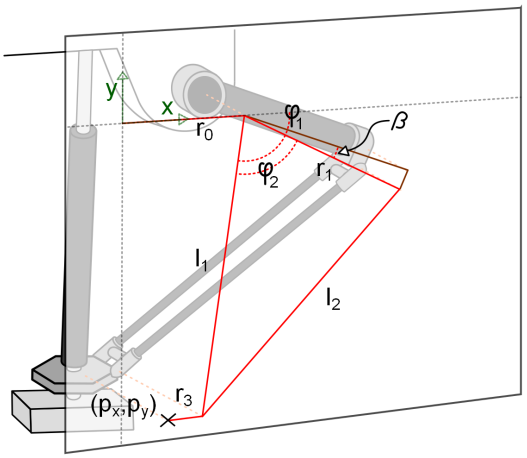

Figure 2.3 This figure shows the geometry, see Fig. 2.2, projected to the axis rotational plane.

$$\cos(\phi_2) = \frac{l_2^2 - l_1^2 - r_1^2}{2 r_1 l_1} \tag{2.3}$$

and the axis angle ϕ_{axis} is given by

$$\phi_{axis} = \phi_1 - \phi_2 - \beta \tag{2.4}$$

The same procedure can be used for axis 2 and 3 by just transforming (p_x, p_y, p_z) to a base which is rotated around y such that its x vector points towards the new axis.

2.3 Rigid-body dynamics

In order to reduce position errors and increase speed it is beneficial to feedforward the torque to the axis servos. The control loops of a servo are based on acting on control errors, which for example means that to compensate for a disturbance it first has to result in a position or velocity error. When feedforward is used the disturbances which are included in the feedforward model are compensated for straight away without creating errors. This reduces the amount of disturbances to be handled by the control loops and therefore increases the accuracy of the servos.

Since the position, speed and acceleration are known both for the travelling plate (in Cartesian coordinates) and for each axes (in motor angles) the torque on each axis can be determined by some basic mechanics.

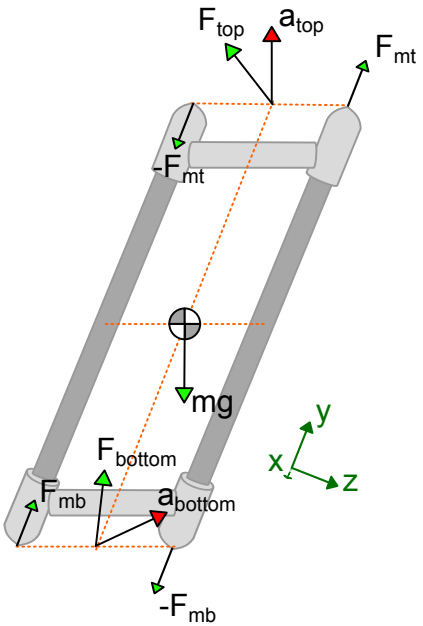

Figure 2.4 This figure shows the lower arm with its forces and accelerations. The green arrows to the right show the coordinate axes.

In Fig. 2.4 the lower arm with its forces and accelerations can be seen. a_{top} is the acceleration at the end of the upper arm and a_{bottom} is the same as the acceleration of the travelling plate. Since the lower arm is a rigid body the two accelerations are given by the acceleration a_m at the arm's centre of mass and the angular acceleration and velocity at the centre of mass.

$$\begin{aligned} a_{top} &= a_m - \tfrac{r_2}{2}(\dot{\theta_z}^2 + \dot{\theta_x}^2)e_y - \tfrac{r_2}{2}\ddot{\theta}_z e_x + \tfrac{r_2}{2}\ddot{\theta}_x e_z \\ a_{bottom} &= a_m + \tfrac{r_2}{2}(\dot{\theta_z}^2 + \dot{\theta_x}^2)e_y + \tfrac{r_2}{2}\ddot{\theta}_z e_x - \tfrac{r_2}{2}\ddot{\theta}_x e_z \\ &\Downarrow \\ a_m &= \frac{a_{top} + a_{bottom}}{2} \\ \ddot{\theta}_x &= \tfrac{2}{r_2} a_{top_z} \\ \ddot{\theta}_y &= 0 \\ \ddot{\theta}_z &= -\tfrac{2}{r_2} a_{top_x} \end{aligned} \tag{2.5}$$

where $\ddot{\theta}_x$, $\ddot{\theta}_y$, $\ddot{\theta}_z$ are the angular accelerations around the coordinate axes and $\dot{\theta}_x$,

$\dot{\theta}_y$, $\dot{\theta}_z$ are the angular velocities.

The forces F_{top} and F_{bottom} in Fig. 2.4 are the sums of the forces at the arm endpoints and the forces F_{mb}, $-F_{mb}$, $-F_{mt}$ and $-F_{mt}$ generates the same torque as the original forces. Since the upper arm (which in Fig. 2.4 would be connected to the top) only can rotate in its plane, forces that generate torque around any other axis then the normal vector to this plane will be cancelled by normal forces. Therefore the torque generated by F_{mb} will be cancelled by F_{mt}. This means that further calculations can view the parallel arms as one rigid body without any external torque applied to it.

$$\begin{aligned} ma_{mx} &= F_{bottomx} + F_{top_x} + mg_x \\ ma_{my} &= F_{bottomy} + F_{top_y} + mg_y \\ ma_{mz} &= F_{bottomz} + F_{top_z} + mg_z \\ I_x\ddot{\theta}_x &= -\tfrac{r_2}{2}F_{bottomz} + \tfrac{r_2}{2}F_{top_z} \\ I_z\ddot{\theta}_z &= \tfrac{r_2}{2}F_{bottomx} - \tfrac{r_2}{2}F_{top_x} \\ &\Downarrow \end{aligned} \tag{2.6}$$

$$\begin{aligned} F_{bottomx} &= \tfrac{1}{2}(ma_{mx} + \tfrac{2}{r_2}I_z\ddot{\theta}_z - mg_x) \\ F_{bottomz} &= \tfrac{1}{2}(ma_{mz} - \tfrac{2}{r_2}I_x\ddot{\theta}_x - mg_z) \\ F_{top_x} &= \tfrac{1}{2}(ma_{mx} - \tfrac{2}{r_2}I_z\ddot{\theta}_z - mg_x) \\ F_{top_y} &= ma_{my} - mg_y - F_{bottomy} \\ F_{top_z} &= \tfrac{1}{2}(ma_{mz} + \tfrac{2}{r_2}I_x\ddot{\theta}_x - mg_z) \end{aligned} \tag{2.7}$$

As can be seen in Eq. (2.7) there are six unknown and five equations which means that $F_{bottomy}$ can be viewed as a free variable. To determine it the forces on the travelling plate have to be determined and to do so one must also find the force contribution from the middle rotational axis.

The forces on the middle rotational axis can be seen in Fig. 2.5. Because of the normal force acting on the top part of the middle rotational axis, only the moment from this part will affect F_{middle}. In x- and z-directions this is also true for the bottom part. But since the bottom part is mounted as a telescopic arm, the y-direction has no normal force component. The length l_m in the following equation is the total length of the middle rotational axis, a_{middle} is written as a_m and F_{middle} is written as F_m.

$$\begin{aligned} \ddot{\theta}_x &= -\frac{1}{l_m}a_{mz} \\ \ddot{\theta}_y &= 0 \\ \ddot{\theta}_z &= \frac{1}{l_m}a_{mx} \end{aligned} \tag{2.8}$$

$$\begin{aligned} I_x\ddot{\theta}_x &= -l_mF_{m_z} - l_tm_tg_z - (l_m - l_b)m_bg_z \\ I_z\ddot{\theta}_z &= l_mF_{mx} + l_tm_tg_x + (l_m - l_b)m_bg_x \\ m_ba_{my} &= F_{my} + m_bg_y \\ &\Downarrow \end{aligned} \tag{2.9}$$

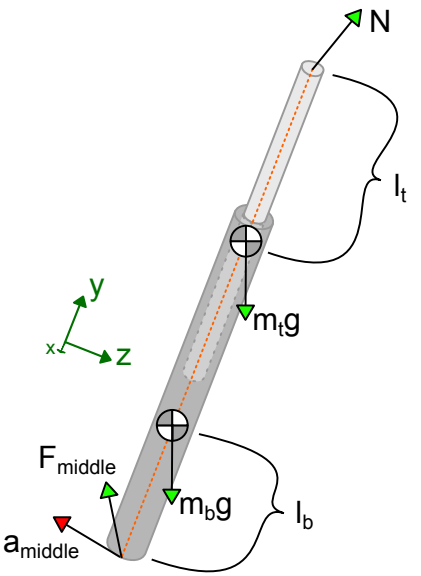

Figure 2.5 Shows the forces on the middle rotational axis and its acceleration.

$$\begin{aligned} F_{mx} &= \frac{1}{l_m}(I_z \frac{1}{l_m} a_{mx} - l_t m_t g_x - (l_m - l_b) m_b g_x) \\ F_{my} &= m_b a_{my} - m_b g_y \\ F_{mz} &= \frac{1}{l_m}(I_x \frac{1}{l_m} a_{mz} - l_t m_t g_z - (l_m - l_b) m_b g_z) \end{aligned} \tag{2.10}$$

Now let's look at the forces on the travelling plate. The torque and angular acceleration can be ignored since the travelling plate's rotation is fixed by the torque from the lower arms. $-\mathbf{F'_m}$ is the force from the middle rotational axis transformed to the travelling plates coordinate system. $-F_1$, $-F_2$ and $-F_3$ are the forces from the three lower arms and e_1, e_2 and e_3 are the three corresponding coordinate bases (in the travelling plate's coordinate system).

$$m\mathbf{a} = m\mathbf{g} - \mathbf{F'_m} - (\textstyle\sum_{i=1}^{3} F_{ix}\mathbf{e_{ix}} + F_{iy}\mathbf{e_{iy}} + F_{iz}\mathbf{e_{iz}}) \tag{2.11}$$

can be written as a linear equation system

$$A\mathbf{x} = \mathbf{b} \tag{2.12}$$

where

$$A = \begin{pmatrix} \mathbf{e_{1y}} & \mathbf{e_{2y}} & \mathbf{e_{3y}} \end{pmatrix} \tag{2.13}$$

$$\mathbf{b} = -m\mathbf{a} + m\mathbf{g} - \mathbf{F'_m} - (\textstyle\sum_{i=1}^{3} F_{ix}\mathbf{e_{ix}} + F_{iz}\mathbf{e_{iz}}) \tag{2.14}$$

which has the solution

$$\mathbf{x} = \begin{pmatrix} F_{1y} \\ F_{2y} \\ F_{3y} \end{pmatrix} = A^{-1}\mathbf{b} \tag{2.15}$$

These forces $F_{bottom_y} = F_{iy}$ can now be inserted into Eq. (2.7) which then yields the F_{top} forces. With these forces determined it is possible to calculate the final torque on the upper arm, see Fig. 2.6.

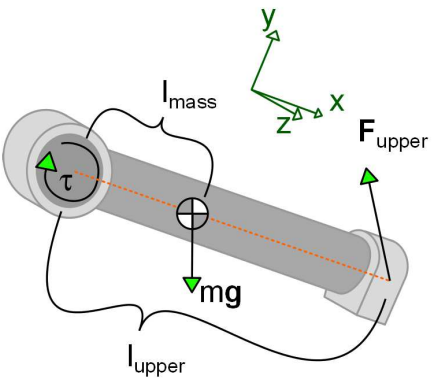

Figure 2.6 This figure shows forces and moment on the upper arm.

$$\mathbf{F}_{upper} = -\mathbf{F'}_{top} \tag{2.16}$$

where $\mathbf{F'}_{top}$ is $\mathbf{F}_{top}$ transformed into the upper arm's coordinate system. The angular acceleration around the axis is then given by

$$\begin{aligned} I\ddot{\theta}_z &= l_{mass}mg_y + l_{upper}F_{upper_y} - \tau \\ &\Downarrow \\ \tau &= l_{mass}mg_y + l_{upper}F_{upper_y} - I\ddot{\theta}_z \end{aligned} \tag{2.17}$$

2.4 Moment of inertia estimation

From pendulum frequency

The resonance frequency of a pendulum can be used to determine its moment of inertia. Without friction the oscillation is governed by the following differential equation

$$I\ddot{\theta} = -l_{mass} \cdot mg \sin(\theta) \tag{2.18}$$

For small angles this has the following solution

$$\theta = A \sin(\frac{\sqrt{l_{mass} \cdot mg}}{\sqrt{I}} t) + B \cos(\frac{\sqrt{l_{mass} \cdot mg}}{\sqrt{I}} t) \tag{2.19}$$

which means that the eigenfrequency is

$$\omega_e \approx \frac{\sqrt{l_{mass} \cdot mg}}{\sqrt{I}} \tag{2.20}$$

For low friction ω_e can be estimated by measuring the time of a number of oscillations

$$\omega_e = 2\pi \frac{N}{T} \tag{2.21}$$

where T is the total time and N the number of oscillations during that time. Rewriting Eq. 2.20 leads to the following

$$I \approx \frac{l_{mass}\, mg}{\omega_e^2} \tag{2.22}$$

If the moment of inertia around the centre of mass is desired then the inertia from the centre of mass has to be removed according to Steiner's theorem.

$$I_{center} \approx \frac{l_{mass}\, mg}{\omega_e^2} - m l_{mass}^2 \tag{2.23}$$

From oscillation torque

For an angular oscillation with given amplitude and frequency the torque is well known and proportional to the moment of inertia around the rotational axis. This can be used for estimating the oscillating object's moment of inertia. The torque and the oscillation have the following dependence

$$\theta = A \sin(\omega t) \tag{2.24}$$

$$\Downarrow$$

$$I\ddot{\theta} = -IA\omega^2 \sin(\omega t) = \tau(t) \tag{2.25}$$

$$\Downarrow$$

$$I = -\frac{\tau(t)}{A\omega^2 \sin(\omega t)} \tag{2.26}$$

Since the torque measurement might be affected by friction and other disturbances one should calculate the inertia at a number of points and then take the mean value to get a good estimate. Around the points $\omega t = \pi \cdot n$ the calculated inertia is extra sensitive to disturbances since the part of $\tau(t)$ which comes from the oscillation becomes very small. Therefore only inertias calculated for points such that

$$|\sin(\omega t)| \geq \alpha, \;\; 0 < \alpha < 1 \tag{2.27}$$

should be used when calculating the mean value.

2.5 System identification

Consider the linear regression model with noise in Eq. (2.28) [7]

$$y_{k+1} = \phi_k^T \theta + w_{k+1}, \quad \phi_k = \begin{bmatrix} y_k \\ \vdots \\ y_{k-n+1} \\ u_k \\ \vdots \\ u_{k-n+1} \end{bmatrix}, \quad \theta = \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{2.28}$$

where y_k and u_k is the system output and input at sample index k, respectively, and w_{k+1} is the noise present in the next output. a and b are model parameters and n is the model order.

If $\{w_k\}_{k=0}^{\inf}$ is white noise, such that $E[w_{k+1} \cdot w_k] = 0$ then θ can be estimated using the least-squares method [7, p. 73]

$$\hat{\theta} = (\Phi_N^T \Phi_N)^{-1} \Phi_N^T Y_N, \quad \Phi_N = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_{N-1}^T \end{bmatrix}, \quad Y_N = \begin{bmatrix} y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{2.29}$$

However, if the noise is not white then the noise part in Φ_N will start to correlate with Y_n, see Eq. (2.29). This makes the estimate biased. This in turn forces the model order to be higher in order to fit the data. One possible way to remove this bias is to use the instrumental variable method [7, p. 113]

$$\hat{\theta} = (Z_N^T \Phi_N)^{-1} Z_N^T Y_N, \quad \Phi_N = \begin{bmatrix} \phi_j^T \\ \vdots \\ \phi_{N-1}^T \end{bmatrix} x, \quad Y_N = \begin{bmatrix} y_{j+1} \\ \vdots \\ y_N \end{bmatrix}, \quad Z_N = \begin{bmatrix} z_j^T \\ \vdots \\ z_{N-1}^T \end{bmatrix} \tag{2.30}$$

with the extra conditions

$$\det(Z_N^T \Phi_N) \neq 0, \quad E[Z_N^T W] = 0, \quad W = \begin{bmatrix} w_j \\ \vdots \\ w_{N-1} \end{bmatrix} \tag{2.31}$$

A standard approach for generating the instrumental variable Z_N is to first estimate a high-order model of the system, and then use that estimate on a new set of data

$$\begin{aligned} Z_N &= \begin{bmatrix} \phi_j'^T \\ \vdots \\ \phi_{N-1}'^T \end{bmatrix} \\ \phi_k' &= \begin{bmatrix} z_k \\ \vdots \\ z_{k-n+1} \\ u_k \\ \vdots \\ u_{k-n+1} \end{bmatrix} \\ z_k &= y_k, \; k \leq n \\ z_{k+1} &= \phi_k'^T \hat{\theta}, \; k > n \end{aligned} \tag{2.32}$$

2.6 Time-optimal in forward and backward time

In order for the patched path, of the forward and the backward paths, to be time-optimal the separate paths have to be time-optimal. In this section it will be shown that the control signal that maximizes the distance from the initial position is the time-optimal for reaching any point (if the torque and velocity at that point is unspecified). It will also be showed that the maximal control signal will maximize the distance, at any given time.

Consider the following continuous-time system

$$\begin{aligned} &\dot{\theta} = \omega \\ &\dot{\omega} = I_{inertia}(\tau + \tau_{grav}) \\ &\dot{\tau} = u \\ &\omega_{min} \leq \omega \leq \omega_{max} \\ &\tau_{min} \leq \tau \leq \tau_{max} \\ &u_{min} \leq u \leq u_{max} \\ &I_{inertia} > 0 \end{aligned} \tag{2.33}$$

with these restrictions on the limits

$$\begin{aligned} &\omega_{min} < 0 \\ &\omega_{max} > 0 \\ &\tau_{min} < -\tau_{grav} \\ &\tau_{max} > -\tau_{grav} \\ &u_{min} < 0 \\ &u_{max} > 0 \end{aligned} \tag{2.34}$$

Let $u(t)$ be any control signal satisfying the limits in Eq. (2.33)

$$\begin{aligned} \tau(t_f) &= \int_{t_0}^{t_f} u(t)dt + \tau(t_0) \\ \omega(t_f) &= I_{inertia} \int_{t_0}^{t_f} \tau(t) + \tau_{grav} dt + \omega(t_0) \\ \theta(t_f) &= \int_{t_0}^{t_f} \omega(t)dt + \theta(t_0) \end{aligned} \tag{2.35}$$

Now let us consider the control signal $u_{pmax}(t)$ which is equal to u_{max} as long as the torque and velocity limits will not be exceeded by using this value. Due to the following integral inequality, [9, p. 292]

$$\begin{gathered} f_0(s) \leq f_1(s) \leq f_2(s) \; \forall \; 0 < s < t \\ \Downarrow \\ \int_0^t f_0(s)ds \leq \int_0^t f_1(s)ds \leq \int_0^t f_2(s)ds \end{gathered} \tag{2.36}$$

the following relationships hold before the torque and velocity limits need to be considered

$$\begin{aligned} u(t) &\leq u_{pmax}(t) = u_{max}, \quad t_0 \leq t \leq t_1 \\ &\Downarrow \\ \tau(t) &\leq \tau_{pmax}(t) = \int_{t_0}^t u_{max} ds + \tau_{pmax}(t_0) \\ &\Downarrow \\ \omega(t) &\leq \omega_{pmax}(t) = I_{inertia} \int_{t_0}^t \tau_{pmax}(s) + \tau_{grav} ds + \omega_{pmax}(t_0) \\ &\Downarrow \\ \theta(t) &\leq \theta_{pmax}(t) = \int_{t_0}^t \omega_{pmax}(s)ds + \theta_{pmax}(t_0) \end{aligned} \tag{2.37}$$

where the time t_1 satisfies either

$$\tau_{pmax}(t_1) = \tau_{max} \tag{2.38}$$

or

$$\begin{aligned} \omega_{pmax}(t_1 + t_b) &= \omega_{max} \\ \tau_{pmax}(t_1 + t_b) &= -\tau_{grav} \end{aligned} \tag{2.39}$$

where t_b is the time it takes to lower $\tau(t_1)$ to reach $\dot{\omega} = 0$.

If the torque limit is reached first, see Eq. (2.38), then together with Eq. (2.37) the following is true

$$\begin{array}{rcl} & u_{pmax}(t) = 0, \quad t_1 < t \leq t_2 & \\ \tau(t) \leq & \tau_{pmax}(t) = \tau_{max} & \\ & \Downarrow & \\ \omega(t) \leq & \omega_{pmax}(t) = I_{inertia} \int_{t_1}^{t} \tau_{pmax}(s) + \tau_{grav} ds + \omega_{pmax}(t_1) & \\ & \Downarrow & \\ \theta(t) \leq & \theta_{pmax}(t) = \int_{t_1}^{t} \omega_{pmax}(s) ds + \theta_{pmax}(t_1) & \end{array} \tag{2.40}$$

where the time t_2 satisfies

$$\begin{array}{c} \omega_{pmax}(t_2 + t_b) = \omega_{max} \\ \tau_{pmax}(t_2 + t_b) = -\tau_{grav} \end{array} \tag{2.41}$$

If the velocity limit would be reached unless the torque is lowered by applying u_{min}, either after the torque limit has been reached or before then

$$\begin{array}{c} t_3 = t_1 \text{ or } t_2 \\ \omega_{pmax}(t_3 + t_b) = \omega_{max} \\ \tau_{pmax}(t_3 + t_b) = -\tau_{grav} \end{array} \tag{2.42}$$

Together with Eqs. (2.37) and (2.40) it is then given that

$$\begin{array}{rcl} \tau(t_3) & \leq & \tau_{pmax}(t_3) \\ \omega(t_3) & \leq & \omega_{pmax}(t_3) \\ \theta(t_3) & \leq & \theta_{pmax}(t_3) \end{array} \tag{2.43}$$

$$\begin{array}{rcl} u_{pmax}(t) & = & u_{min}, \quad t_3 < t \leq t_3 + t_b \\ \tau_{pmax}(t) & = & \int_{t_3}^{t} u_{pmax}(t) ds + \tau_{pmax}(t_3) \\ \omega_{pmax}(t) & = & I_{inertia} \int_{t_3}^{t} \tau_{pmax}(s) + \tau_{grav} ds + \omega_{pmax}(t_3) \\ \theta_{pmax}(t) & = & \int_{t_3}^{t} \omega_{pmax}(s) ds + \theta_{pmax}(t_3) \end{array} \tag{2.44}$$

Putting together Eqs. (2.42) and (2.51) gives

$$\omega_{pmax}(t_3 + t_b) = I_{inertia} \int_{t_3}^{t_3 + t_b} \tau_{pmax}(s) + \tau_{grav} ds + \omega_{pmax}(t_3) = \omega_{max} \tag{2.45}$$

$$\begin{array}{rcl} \tau_{pmax}(t_3 + t_b) & = & \int_{t_3}^{t_3 + t_b} u_{min} ds + \tau_{pmax}(t_3) = -\tau_{grav} \\ & \Downarrow & \\ \int_{t_3}^{t_3 + t_b} u_{min} ds & = & u_{min} t_b = -\tau_{pmax}(t_3) - \tau_{grav} \\ & \Downarrow & \\ t_b & = & -\dfrac{\tau_{pmax}(t_3) + \tau_{grav}}{u_{min}} \end{array} \tag{2.46}$$

Now let us assume that there exists a time t such that

$$\omega(t) > \omega_{pmax}(t), \;\; t_3 < t \leq t_3 + t_b \tag{2.47}$$

Since $\omega(t_3) \leq \omega_{pmax}(t_3)$, see Eq. (2.43), and $\omega(t) \in C^1, \; \tau(t) \in C^0 \; \forall \, t$ this leads to that there exists a t_i such that

$$\begin{array}{c} \omega(t_i) = \omega_{pmax}(t_i) \\ \tau(t_i) > \tau_{pmax}(t_i) \end{array} \tag{2.48}$$

Then, for $t_i \leq t \leq t_3 + t_b$ the following holds

$$\begin{array}{rcl} \tau(t) & = & \int_{t_i}^{t} u(s)ds + \tau(t_i) \geq \\ & \geq & \int_{t_i}^{t} u_{min}ds + \tau(t_i) > \\ & > & \int_{t_i}^{t} u_{min}ds + \tau_{pmax}(t_i) = \tau_{pmax}(t) \end{array} \tag{2.49}$$

This leads to

$$\begin{array}{rcl} \omega(t_3 + t_b) & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega(t_i) = \\ & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega_{pmax}(t_i) > \\ & > & \int_{t_i}^{t_3+t_b} \tau_{pmax}(s)ds + \omega_{pmax}(t_i) = \\ & = & \omega_{pmax}(t_3 + t_b) = \omega_{max} \end{array} \tag{2.50}$$

So $\omega(t) > \omega_{pmax}(t), \;\; t_3 < t \leq t_3 + t_b$ cannot be true without $\omega(t_3 + t_b) > \omega_{max}$ and since this result violates the limits, $\omega(t)$ cannot be greater than $\omega_{pmax}(t)$. This leads to

$$\begin{array}{c} u_{pmax}(t) = u_{min}, \;\; t_3 < t \leq t_3 + t_b, \;\; t_b = -\dfrac{\tau_{pmax}(t_3) + \tau_{grav}}{u_{min}} \\ \begin{array}{rcl} \omega(t) & \leq & \omega_{pmax}(t) \\ & \Downarrow & \\ \theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \leq & \theta_{pmax}(t) = \int_{t_3}^{t} \omega_{pmax}(s)ds + \theta_{pmax}(t_3) \end{array} \end{array} \tag{2.51}$$

For $t > t_3 + t_b$

$$\begin{array}{rcl} \omega(t) & \leq & \omega_{pmax}(t) = \omega_{max} \\ & \Downarrow & \\ \theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \leq & \theta_{pmax}(t) = \int_{t_3}^{t} \omega_{pmax}(s)ds + \theta_{pmax}(t_3) \end{array} \tag{2.52}$$

Combining Eqs. (2.37), (2.40), (2.51) and (2.52) the following inequalities are yielded

$$\begin{array}{rcl} \omega(t) & \leq & \omega_{pmax}(t) \\ \theta(t) & \leq & \theta_{pmax}(t) \end{array} \quad \text{for} \;\; t \geq t_0 \tag{2.53}$$

Now consider three arbitrary state values which do not violate any limits

$$\begin{matrix} \theta_0 \\ \omega_0 \\ \tau_0 \end{matrix} \tag{2.54}$$

and that at $t = t_0$

$$\begin{matrix} \theta(t_0) = \theta_{pmax}(t_0) = \theta_0 \\ \omega(t_0) = \omega_{pmax}(t_0) = \omega_0 \\ \tau(t_0) = \tau_{pmax}(t_0) = \tau_0 \end{matrix} \tag{2.55}$$

where $\theta_{pmax}(t_0)$ is the maximum system response deduced previously.

Now say that there exists a time-optimal control signal $u(t)$ (the control signal for $\theta(t)$) that yields

$$\theta(t_f) = \theta_f > \theta_0 \tag{2.56}$$

Then it follows from Eq. (2.53) that

$$\theta_{pmax}(t_f) \geq \theta(t_f) = \theta_f > \theta_0 = \theta_{pmax}(t_0) \tag{2.57}$$

Since $\theta_{pmax}(t)$ and $\theta(t)$ are continuous ($\theta_{pmax}(t) \in C^2,\ \theta(t) \in C^2$) there exists a $t \leq t_f$ such that

$$\theta_{pmax}(t) = \theta_f \tag{2.58}$$

which means that $u_{pmax}(t)$ is a time-optimal solution for reaching $\theta_f > \theta_0$.

For values smaller than θ_0 a minimizing control signal $u_{pmin}(t)$, see Appendix 7.2, exists which results in the following inequalities

$$\begin{matrix} \omega(t) & \geq & \omega_{pmin}(t) \\ \theta(t) & \geq & \theta_{pmin}(t) \end{matrix} \quad \text{for } t \geq t_0 \tag{2.59}$$

and similarly to the previous case

$$\theta_{pmin}(t_f) \leq \theta(t_f) = \theta_f < \theta_0 = \theta_{pmin}(t_0) \tag{2.60}$$

which, since also $\theta_{pmin}(t) \in C^2$, proves the existence of a $t \leq t_f$ such that

$$\theta_{pmin}(t) = \theta_f \tag{2.61}$$

And therefore $u_{pmin}(t)$ is a time-optimal control for $\theta_f < \theta_0$

The backward time case is very similar to these two as well. Since the only difference between an integral

$$\int_{t_0}^{t_1} f(t)dt,\ \ t_1 > t_0 \tag{2.62}$$

and one with $t_1 < t_0$ is that the sign changes [9, p. 292]

$$\int_{t_0}^{t_1} f(t)dt = -\int_{t_1}^{t_0} f(t)dt \tag{2.63}$$

For negative integral times this means that the integral rule in Eq. (2.36) is changed to

$$\begin{array}{c} f_0(s) \leq f_1(s) \leq f_2(s) \; \forall \; t < s < 0 \\ \Downarrow \\ \int_0^t f_2(s)ds \leq \int_0^t f_1(s)ds \leq \int_0^t f_0(s)ds \end{array} \tag{2.64}$$

Now let's introduce the control signal $u_{nmin}(t)$. In backward time this signal is equal to u_{max} as long as the torque and velocity limits are inactive and this yields the following inequalities

$$\begin{array}{rcl} u(t) & \leq & u_{nmin}(t) = u_{max}, \quad t_1 \leq t \leq t_0 \\ & \Downarrow & \\ \tau(t) & \geq & \tau_{nmin}(t) = \int_{t_0}^{t} u_{max}ds + \tau_{nmin}(t_0) \\ & \Downarrow & \\ \omega(t) & \leq & \omega_{nmin}(t) = I_{inertia} \int_{t_0}^{t} \tau_{nmin}(s) + \tau_{grav}ds + \omega_{nmin}(t_0) \\ & \Downarrow & \\ \theta(t) & \geq & \theta_{nmin}(t) = \int_{t_0}^{t} \omega_{nmin}(s)ds + \theta_{nmin}(t_0) \end{array} \tag{2.65}$$

By continuing with the same approach as before to handle the torque and velocity limits the following inequalities are are yielded (see Appendices 7.4 for the whole derivation)

$$\begin{array}{rcl} \omega(t) & \leq & \omega_{nmin}(t) \\ \theta(t) & \geq & \theta_{nmin}(t) \end{array} \quad \text{for } \; t \leq t_0 \tag{2.66}$$

The hole derivation of the backwards time maximizing control signal $u_{nmax}(t)$ can also be found in Appendix 7.3 and Eq. (2.67) shows the resulting inequalities.

$$\begin{array}{rcl} \omega(t) & \geq & \omega_{nmax}(t) \\ \theta(t) & \leq & \theta_{nmax}(t) \end{array} \quad \text{for } \; t \leq t_0 \tag{2.67}$$

By the same reasoning as before with $u_{pmax}(t)$, Eqs. 2.66 and 2.67 leads to that $u_{nmin}(t)$ is the time-optimal solution for $\theta_f < \theta_0$ and $u_{nmax}(t)$ for $\theta_f > \theta_0$. Worth pointing out again here is that the time is inverted, so θ_f will be reached at an earlier point in time than when the system starts in θ_0.

Let us now say that there is a start state vector, a finish vector and the control signal to go from start to finish in minimum-time is desired. According to the Bellman principle of optimality [3], each component of the optimal-time solution has to be time-optimal with respect to its initial condition.

If one goes forward from the start states and backward from the finish states, as discussed in the introduction, and there exists a point in between where the states

are equal then the two different control sequences can be patched together. Since the two cases are time-optimal for reaching this intersecting point the total control sequence becomes time-optimal.

In the car example case discussed in the introduction this works because that system only has two states. Only two variables are free to solve for, the forward and backward times up to the intersection, and this means that only two equations can be solved. In the case with jerk restrictions there are three states and hence an intersection between forward and backward cannot be guaranteed.

Let us assume that there exists a time-optimal control signal $u_{sync}(t)$ that is inserted in between the forward and backward case that guarantees intersection. Since there are three states (which lead to three equations) and two variables apart from the intermediate it is enough for the synchronization case to yield equality for the torque state. The torque is given by the following

$$\begin{gathered} \tau_{backward} = \int_0^{t_{sync}} u_{sync}(s+t_{forward})ds + \tau_{forward} \\ \Downarrow \\ \int_0^{t_{sync}} u_{sync}(s+t_{forward}) = \tau_{backward} - \tau_{forward} \end{gathered} \tag{2.68}$$

$$\int_0^{t_{sync}} u_{min} \leq \int_0^{t_{sync}} u_{sync}(s+t_{forward}) \leq \int_0^{t_{sync}} u_{max} \tag{2.69}$$

As before this gives that the time-optimal control signal is, without considering the torque and velocity limits

$$u_{sync}(t) = \begin{cases} u_{min}, & \tau_{backward} - \tau_{forward} < 0 \\ \\ u_{max}, & \tau_{backward} - \tau_{forward} > 0 \end{cases} \tag{2.70}$$

where $t_{forward} \leq t \leq t_{forward} + t_{sync}$. This leads to

$$u_{sync}(t) = \begin{cases} \tau_{backward} = u_{min} t_{sync} + \tau_{forward}, & \tau_{backward} - \tau_{forward} < 0 \\ \\ \tau_{backward} = u_{max} t_{sync} + \tau_{forward}, & \tau_{backward} - \tau_{forward} > 0 \end{cases} \tag{2.71}$$

and

$$u_{sync}(t) = \begin{cases} t_{sync} = \dfrac{\tau_{backward} - \tau_{forward}}{u_{min}}, & \tau_{backward} - \tau_{forward} < 0 \\ \\ t_{sync} = \dfrac{\tau_{backward} - \tau_{forward}}{u_{max}}, & \tau_{backward} - \tau_{forward} > 0 \end{cases} \tag{2.72}$$

In the case that the final angle is larger than the starting angle the forward cases should use $u_{pmax}(t)$ and the backward case should use $u_{nmin}(t)$, since they have to approach each other. If the synchronization, at the same time, would have the case

$\tau_{backward} - \tau_{forward} > 0$ then it should use u_{max}. This would mean that the control signal, if the torque and velocity limits are inactive, would be the same for forward, sync and backward since all use u_{max}. Effectively this would only be one case with one time variable, which could be handled by just using the forward iterating. So the synchronization control signal must be opposite to the others to serve any purpose. This leads to the following

$$u_{sync}(t) = \begin{cases} u_{min}, & \theta_{start} < \theta_{finish} \\ u_{max}, & \theta_{start} > \theta_{finish} \end{cases} \tag{2.73}$$

Now consider the torque and velocity limits together with $u_{sync}(t)$. Since the $\tau_{forward}$ and $\tau_{backward}$ both are within the limits (due to $u_{pmax}(t)$, $u_{pmin}(t)$, $u_{nmax}(t)$ and $u_{nmin}(t)$), and $\tau_{sync}(t)$ is given by equation 2.71, the synchronization case will not have to consider the torque limits. The velocities $\omega_{forward}$ and $\omega_{backward}$ will also be within the limits, but $\omega_{sync}(t)$ is not guaranteed to be in between them. Since $u_{sync}(t) = u_{min}$ when synchronizing $u_{pmax}(t)$ and $u_{sync}(t) = u_{max}$ when synchronizing $u_{pmin}(t)$ the following inequality holds

$$\begin{aligned} u_{sync}(t) &\leq u_{pmax}(t), \text{ when synchronizing } u_{pmax}(t) \\ u_{sync}(t) &\geq u_{pmin}(t), \text{ when synchronizing } u_{pmin}(t) \end{aligned} \tag{2.74}$$

For the $u_{pmax}(t)$ synchronizing it can then be said

$$\begin{aligned} \omega_{sync}(t) &= I\int_0^t(\int_0^r(u_{min})ds + \tau_{forward} + \tau_{grav})dr + \omega_{forward} \\ &\leq I\int_0^t(\int_0^r(u_{pmax}(s))ds + \tau_{forward} + \tau_{grav})dr + \omega_{forward} \\ &= \omega_{forwardmax}(t) \leq \omega_{max} \end{aligned} \tag{2.75}$$

So $\omega_{sync}(t) \leq \omega_{max}$. Since $\ddot{\omega}_{sync}(t) \propto u_{sync}(t) = u_{min} < 0$ for $t_{forward} \leq t \leq t_{forward} + t_{sync}$ it means that $\omega_{sync}(t)$ may only have one extremum and in case it exists it is a maximum. This means that $\omega_{sync}(t)$ will have its minimum at $t = t_{forward}$ or $t = t_{forward} + t_{sync}$ and at this points it should be equal to $\omega_{forward}$ and $\omega_{backward}$. So the $\omega_{sync}(t)$ cannot exceed the velocity limits with feasible initial conditions.

This means that the synchronizing case for $u_{pmax}(t)$ is always $u_{sync}(t) = u_{min}$ since the torque and velocity limits will not be exceeded. The same approach for the $u_{pmin}(t)$ can be used, see Appendix 7.5, and also in this case the limits can be neglected.

2.7 Backwards time model

A linear continuous-time state-space model has the following form

$$\dot{x} = Ax + Bu \tag{2.76}$$

The time derivative is given by [9, p. 179]

$$\dot{x} = \lim_{dt \to 0^+} \frac{x(t+dt) - x(t)}{dt} \tag{2.77}$$

To change the direction of time for the continuous model the sign of dt has to be changed

$$\begin{aligned} &\lim_{dt \to 0^+} \frac{x(t-dt) - x(t)}{dt} \\ &= \lim_{dt \to 0^+} -\frac{x(t) - x(t-dt)}{dt} \end{aligned} \tag{2.78}$$

This is equivalent with changing the sign of the A and B matrices such that

$$\begin{aligned} A_{back} &= -A \\ B_{back} &= -B \end{aligned} \tag{2.79}$$

Since the A matrix changes sign the backward system changes poles. The poles of the forward system is given by

$$\det(sI - A) = 0 \tag{2.80}$$

where the values of s that solve the equation are the poles. For the backwards system the equation has the following form

$$\begin{aligned} \det(sI - A_{back}) &= \det(sI + A) \\ &= -\det((-s)I - A) = 0 \end{aligned} \tag{2.81}$$

So a stable pole (negative eigenvalue) in the forward system will become unstable in backward time and vice versa.

2.8 Zero-order-hold Sampling of a System

Zero-order hold sampling is an exact technique to discretize a linear system, under the assumption that the inputs to the system are constant in between the sampling times (which is true for most digital systems) [2, p. 36].

Given the system

$$\begin{aligned} \dot{x} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.82}$$

The system can be discretized according to

$$\begin{aligned} x(kh+h) &= A'x(kh) + B'u(kh) \\ y(kh) &= Cx(kh) + Du(kh) \end{aligned} \tag{2.83}$$

where

$$\begin{aligned} A' &= e^{Ah} \\ B' &= \int_0^h e^{As} ds B \end{aligned} \tag{2.84}$$

2.9 State feedback controller

Consider the discrete-time linear state-space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.85}$$

A discrete-time state feedback control law is given by [2, p. 198]

$$u_k = -L \cdot x_k + l_r \cdot r \tag{2.86}$$

By inserting this law into Eq. (2.85) the system becomes

$$\begin{aligned} x_{k+1} &= (A - BL)x_k + Bl_r \cdot r \\ y_k &= (C - DL)x_k + Dl_r \cdot r \end{aligned} \tag{2.87}$$

The poles (eigenvalues) of this system are given by solutions z to the characteristic equation

$$\det(zI - (A - BL)) = 0 \tag{2.88}$$

If the system 2.85 is controllable then L can be chosen such that the poles of the system can be placed at any combination of points. l_r can then be calculated such that the system gets the desired gain from r to y.

2.10 Deadbeat controller

A deadbeat controller is a sub group of discrete-time controllers where all the poles of the system are placed in zero [2, p. 203]. This results in that the system reaches the reference r in at most the same number of time-steps as the states of the system. For any given sampling rate, it is therefore time-optimal when no limits are applied to the system.

2.11 MapleSim

MapleSim is a graphical acausal simulation program [8]. The interface is similar to Simulink in the sense that systems are created by wiring together different blocks.

Since it uses acausal blocks the direction of the wiring does not have to be defined but is determined at execution time. This is very useful for building physical models since energy can go in both directions. A propeller connected to a motor can for example be used for generating wind but it can also be used for generating electricity from wind. In MapleSim the same model can be used in both cases since the motor shaft can be both input and output.

Another benefit with using MapleSim is that a model can be exported into Maple worksheets as a set of equations. These can then be analysed with Maple's extensive library of symbolic tools.

3

Methods

3.1 Inertia estimation

For inertia estimation of the parallel arm and the two arms in de middle arm assembly the resonance frequency approach described in Sec. 2.4 was used. The mass centre position for each arm was estimated by balancing them on a sharp edge of a table and measure the distance between the table edge and the joint of the arm. To determine their mass a scale with 0.005 [kg] resolution was used. The resonance frequency for each arm was then determined by holding the joint of the arm and gently moving back and forth at a frequency that resulted in the highest amplitude. A certain number of periods were performed and the total time was measured with a stopwatch.

The inertia of the upper arm was estimated by the oscillation torque approach described in Sec. 2.4. This was done by sending three sinusoidal signals with the same amplitude but varying frequency as position reference to the servo drive. The measured torques were then used to calculate the inertia at a number of time-points with Eq. (2.26). The estimated inertia for each frequency was then calculated by taking the mean value of the calculated inertias. $\alpha = 0.3$ was used to remove too uncertain inertias, see Eq. (2.27), when calculating the mean value.

3.2 Model of one axis

For creating a model of the Flexpicker servos the documentation of the position and velocity loop of the ACOPOS 1045 drive was used with the current parameters for the setup. The transfer function from current (stator quadratic component) reference (output from the velocity loop) to torque was identified as follows.

A PRBS signal was sent as a torque reference to the servo to excite the system. In order to stabilize and keep the servos movements within the allowed region, the mean of the test signal was shifted depending on the current position, see Eq. (3.1).

$$\tau_{ref} = \tau_{amp} \cdot \tau_{PRBS} - k \cdot (\theta - \theta_{testposition}) \tag{3.1}$$

where τ_{PRBS} is a normalized pseudorandom binary sequence with zero mean.

The torque is proportional to the current (stator quadrature component) [4]. So the identification was done from current reference to actual current output, which then can be scaled based on data from the servo motor to get the actual torque.

The test was conducted with torque amplitudes 0.5 Nm and 0.2 Nm. The sample time was 0.1 ms and the torque reference was updated every 72 samples. The sample acquisition was done using the built in data gathering function on the ACOPOS drive. Three data series for 0.5 Nm amplitude and one for 0.2 Nm were acquired. The two first series for torque amplitude 0.5 Nm were used for estimating model parameters with both the least-squares method and the instrumental variable method. The least-squares method was used on the first data series to generate models of orders from 1 to 4 and one with order 30. The instrumental variable Z_N was generated from the 30-order-model and the second data series. The instrumental variable method was then used to generate models of orders from 1 to 3. All models were validated against both the third data series for torque amplitude 0.5 Nm and the one for amplitude 0.2 Nm.

The second-order instrumental variable estimate was chosen as the best fit. The acquired discrete-time system was transformed to a continuous-time transfer function with the "d2c" function in Matlab using zero-order hold. The transfer function was scaled with "MOTOR_TORQ_CONST" 0.4884Nm/A from the ACOPOS parameter table [13], such that the output was in torque.

The acquired transfer function together with the position and velocity loop schematics were used to build up a symbolic continuous-time block model of the axis in MapleSim. The input to the transfer functions was connected to the current reference from the velocity loop blocks and the output to an inertia block and in between a constant torque block (simulating external forces on the load). This resulted in a model from position reference and torque feedforward to shaft angle with load inertia and external torque as parameters, see Figs 3.1-3.2.

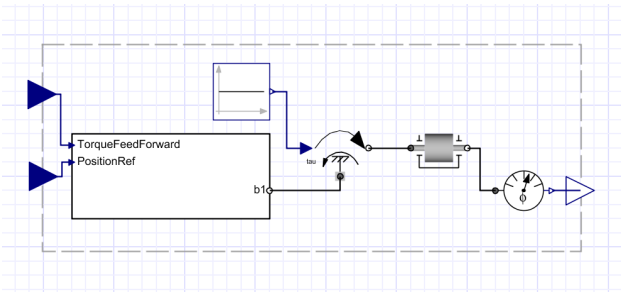

Figure 3.1 Here the model of one axis created in MapleSim can be seen.

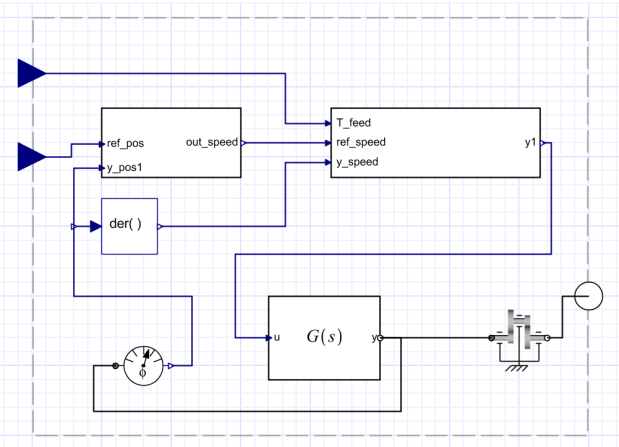

Figure 3.2 Here the subparts of the middle block in figure 3.1 can be seen. $G(s)$ is the torque reference transfer function, the top left block is the position loop and the top right is the velocity loop.

The dynamics of the position and velocity loop act on position and velocity errors. Since the system uses torque feedforward an error may only occur if the model used for feedforward is not the same as the actual load. Because the same system will be used for both forward and backward simulation and torque calculations there will be no position errors for the simulations. Therefore, the dynamics of the position and velocity feedback loops were removed. So the model for simulations was reduced to only include torque to shaft angle. To make it possible to get the model on linear state-space form the saturations were also removed (and added in the forward and backward time step iteration function later on). The simplified and linear model was symbolically imported to Maple from MapleSim as ODEs (Ordinary Differential Equations) via the DynamicSystems package. The ODEs were translated to a continuous-time linear state-space model with the inertia and external torque as symbolic parameters. This state-space model was then discretized with zero-order hold to get the discrete forward model. To generate the backwards time model the sign of the matrices for the forward continuous-time linear state-space model was changed, see theory 2.7. This backward-continuous-time model was then also discretized with zero-order hold.

The model yielded was considered too complex to work with since it became highly unstable in backward time and had too many states. Therefore, the current transfer function was removed and instead an integrator was used. A more detailed explanation of why can found later on in Sec. 4.5. The input to the new model then became the time derivative of the torque. This made it possible to limit both the torque and the time derivative with a model with only three states. The resulting

state-space system for forward and backward time iteration can be seen in Fig. 3.3.

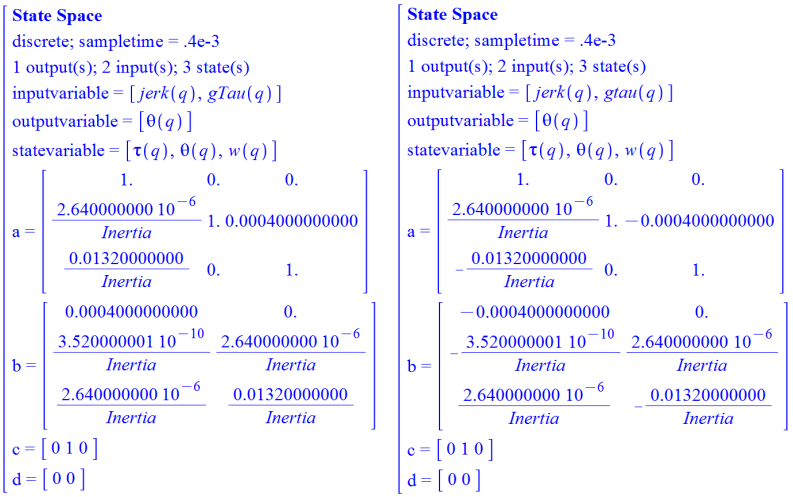

Figure 3.3 Here the discrete-time state-space models used can be seen. The left state-space system is for forward time and it is named "discCurrentSys" in Maple. The right state-space system is for backward time and it is named "backDiscCurrentSys".

Two Maple functions, one for iterating the state vector one time step forward and one for backward (while taking the limits of the system into account), were created based on the $u_{max}(t)$ and $u_{min}(t)$ from Sec. 2.6. The Maple code of the two functions can be seen in Appendix 7.1.

The inputs to the functions consisted of a state vector, a control signal, load moment of inertia and an external torque load. The data returned from the functions were the state vector one time step ahead. In the functions, a test before using the discrete-time state space model matrices was added to determine if the control signal would result in the torque exceeding its limits after iterating. In this case the control signal was replaced with a value that would reach the limit exactly. A test for keeping the velocity limits was also added. This was done by checking how long time, given the control signals limits, it would take to go from the current torque to the torque which keeps the velocity constant. This time t_b, see Sec.2.6, was then used to calculate what the velocity would be at this point. If it exceeded the limits the control signal was replaced with the one that fastest yields a constant velocity. This approach did not take the discreteness of the system into account so the velocity could exceed its limits. To decrease the amount of this problem, the time t_b was increased with h (the sample-time) to make the braking occur earlier. A test for if the control signal exceeded its limits was also added and in the case it did it was set

to the maximal or minimal value.

For all the tests conducted in this thesis the finish state-vector was generated from a trajectory in form of a step with zero velocity and torque at all points. τ_{grav} was set to zero and $I_{inertia} = 0.1Kg \cdot m^2$. The sampling time h was set to 0.4 ms, which was the sample time of the position loop on the ACOPOS drives.

3.3 Aligning velocity and torque states

A Maple function, called "brakingDataFromTauAndW", was created that took the desired finish state vector and the current state vector, see Appendix 7.1 for Maple code. It then returned the angle at which the velocity and torque of the current state and the backwards generated state were equal. This was done by using the backwards iterating function. The control signal given to the backward function was u_{max} if the finish angle was bigger than the current and u_{min} if it was smaller, see Sec. 2.6. After each iteration the torques were synchronized by applying the control signal $u_{sync}(t)$, see Sec. 2.6. The back-stepping and synchronization were put in a loop which continued to iterate backwards and synchronize until the resulting velocity no longer was in between the finish velocity and the current velocity. After the loop was completed the previous synchronized state (that had a velocity that was in between the finish and current velocity) and the latest state were linearly interpolated such that the interpolated state had the same velocity as the current state. The interpolated state was then returned from the function, where the angle state represented the angle at which the velocity and torque were equal to the current states.

3.4 Temporal interpolation approach

This approach of patching together the forward and backward paths was based on finding the approximate time at which the intersection occurs (which may be in between two sample times). The approximate backward path (interpolation of the two closest discrete-time paths) was then sampled at a given time forward from the current sample. The angle of this sample was then sent to a deadbeat controller, see Sec. 2.10, to generate the control signal to send to the forward model. A new approximate backward intersecting path was calculated at each time-step to generate the control signal to send to the forward model to get it to follow the backward path.

For this approach the "brakingDataFromTauAndW" function was modified so that it returned both the backwards generated state vector and a sample of the backward path at a desired time from the velocity intersection. This function was called once with the current state vector and once with a state vector one time-step ahead with u_{max}. This returned four angles, which can be seen in Fig. 3.4, based on the current state vector and the next state vector. The angles θ_c and θ_n, which also can

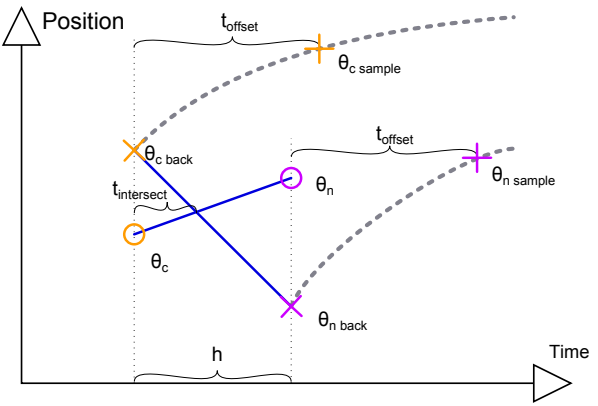

Figure 3.4 Here the angles and times for the"Temporal interpolation" approach can be seen. θ_c is the current angle, $\theta_{c\ back}$ is the backwards generated angle, $\theta_{c\ sample}$ is a sample of the backwards path at t_{offset} from the current time. θ_n, $\theta_{n\ back}$ and $\theta_{n\ sample}$ are the same as the above but for the state vector one time step ahead with either u_{min} or u_{max} as control signal.

be seen in the figure, are the angles of the current and the next state vector. $\theta_{c\ back}$ and $\theta_{n\ back}$ are the angles where the backward paths intersected (in torque and velocity) with the current state vector and the next state vector respectively. The angles in between the sampling points were approximated by linear interpolation. This is represented with one line between θ_c and θ_n and one between $\theta_{c\ back}$ and $\theta_{n\ back}$ in the figure. If the intersection point between the two lines occurred between the current and the next time step then

$$t_{\text{offset}} = t_{\text{sample ahead}} - t_{\text{intersect}} \tag{3.2}$$

where the time $t_{sample\ ahead}$ was a free parameter for this approach. The following interpolation of $\theta_{c\ sample}$ and $\theta_{n\ sample}$ was then sent as reference to the deadbeat controller

$$(1 - \frac{t_{\text{intersect}}}{h})\theta_{\text{c sample}} + \frac{t_{\text{intersect}}}{h}\theta_{\text{n sample}} \tag{3.3}$$

By using this sample offset and interpolation the reference became an approximate sample (at time $t_{sample\ ahead}$ from the current time-step) of the path that intersected at $t_{intersect}$.

If the intersection point between the two lines occurred more than one time step ahead, then the finish angle was sent to the deadbeat controller instead. In the case that the intersection instead occurred before the current time step, then t_{offset} was set to $t_{sample\ ahead}$ and the resulting $\theta_{c\ sample}$ was sent to the deadbeat controller.

3.5 State feedback controlled intersection approach

This intersection method was based on a state feedback controller, see Sec. 2.9, that applied an additional control signal to the one at the end of the backwards generated path to remove the difference between the current state vector and the backwards generated state vector. The value of the control signal at the end of the path was decided by looking at the synchronizing time t_{sync}. If it was smaller than or equal to $-h$ then $u_{sync}(t)$ would be active during the whole next time step. If it was equal to or greater than zero then the control signal used for the last backwards iteration would be active during the whole next time step. In the case that $-h < t_{sync} < 0$ then there would be a switch between the two control signal at some point during the next time step. This case was chosen to be approximated by a linear interpolation between the two control signals.

The state vector returned from the backwards iterating function was an interpolation between two different state vectors, see Sec. 3.2, so the t_{sync} and the $u_{sync}(t)$ used above were interpolations of the data from the two different paths corresponding to the two state vectors.

The backwards generated path and the forward can be written on discrete-time linear state-space form, if $u_{back}(t_k)$ and $Bu_{current}(t_k)$ are assumed to follow the state and control signal limits.

$$\begin{aligned} x_{back}(t_{k+1}) &= Ax_{back}(t_k) + Bu_{back}(t_k) \\ x_{current}(t_{k+1}) &= Ax_{current}(t_k) + Bu_{current}(t_k) \end{aligned} \tag{3.4}$$

The error between $x_{back}(t_{k+1})$ and $x_{current}(t_{k+1})$ is then given by

$$\begin{aligned} & x_{back}(t_{k+1}) - x_{current}(t_{k+1}) = \\ = \;& A(x_{back}(t_k) - Ax_{current}(t_k)) + B(u_{back}(t_k) - u_{current}(t_k)) = \\ = \;& \widetilde{x}(t_{k+1}) = A\widetilde{x}(t_k) + B\widetilde{u}(t_k) \end{aligned} \tag{3.5}$$

By using state feedback control

$$\widetilde{u}(t_k) = L \cdot \widetilde{x}(t_k) = u_{back}(t_k) - u_{current}(t_k) \tag{3.6}$$

The poles of the error dynamics can be placed arbitrarily (since the system is controllable). The pole placement was done according to three free parameters representing the time constants of three continuous-time poles. The poles were given by $s_{pole} = -1/\lambda_{time}$ and together with the formula [2, p. 56]

$$z_{pole} = e^{h \cdot s_{pole}} \tag{3.7}$$

The discrete poles z_{pole} were calculated via the formula

$$e^{-h/\lambda} \tag{3.8}$$

From the given pole placement L was calculated, and the corresponding $u_{current}(t_k)$ was given by

$$u_{current}(t_k) = u_{back}(t_k) - L \cdot \widetilde{x}(t_k) \tag{3.9}$$

The state feedback controller has no recollection of the limits on the control signal so in order to give $\widetilde{u}(t_k)$ room to work without saturating $u_{current}(t_k)$, the control signal limits used by the backwards iterating function were set to 90 % of the maximum.

As a compliment to the above described method a deadbeat controller was implemented, see Sec. 2.10. By using the current state vector and the finish angle the control signals from the deadbeat controller for the next three time steps were generated. If all three control signals were within u_{max} and u_{min} then it was assumed that the control-sequence from the deadbeat was feasible and its first control signal was sent to the forward iterating function. If one of them were outside of the range then the $u_{current}(t_k)$ was sent to the forward iterating function instead.

3.6 Intersection by control signal approach

This intersection method was based on finding the control signal that makes the state vector, at a given time ahead, equal to the backwards generated state vector and then apply the first sample of that control signal. This was done by testing different control sequences. Two different types of sequences were tested. One that was just one sample long with an arbitrary test value v and another which was two samples long with v during the first sample and the second set to zero

Type-1:

$$u_k = v \tag{3.10}$$

Type-2:

$$\begin{aligned} u_k &= v \\ u_{k+1} &= 0 \end{aligned} \tag{3.11}$$

In both of the two sequence cases, 21 equally spaced values on v, between u_{min} and u_{max}, were used to generate test state vectors (iterated forward from the current states). Each generated state vector was sent to "brakingDataFromTauAndW", see Sec. 3.2, to get the corresponding backwards generated angle. The resulting data for five tests could look something like Fig. 3.5.

By doing linear interpolation in between each backwards generated angle (blue cross) and each corresponding test angle (orange circle) an intersection for each data pair (two neighbouring test values) was generated. If the intersection was in between one of the pairs then that value was used and sent to the forward iterating function as control signal. If no intersections occur in between any of the pairs, additional tests were performed. If there existed pairs with negative intersection then the intersection of the pair first from the left was chosen.

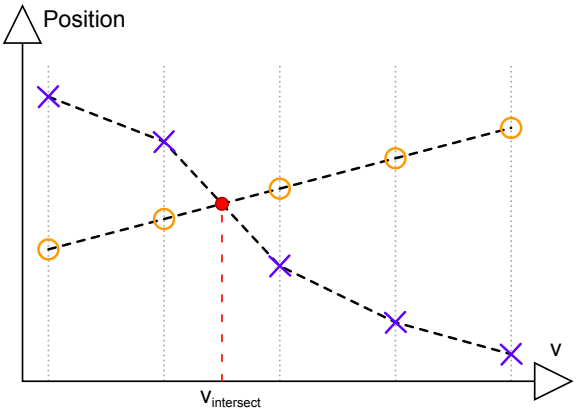

Figure 3.5 Orange circles represent the angle of the state vector after applying the control sequence with the given test value v. The blue crosses are the corresponding angle of the backwards generated state vector. The red dot represents the intersection point with $v_{intersect}$.

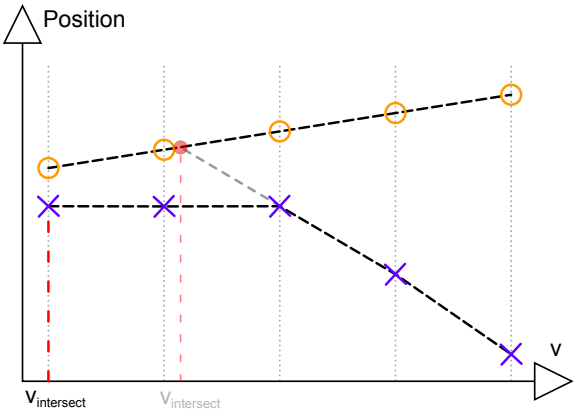

Figure 3.6 This is an example of the case where no intersection occurs in between the pairs (in the case when the finish angle was larger than the current). Orange circles represent the angle of the state vector after applying the control sequence with the given test value v. The blue crosses are the corresponding angles of the backwards generated state vectors. In this case the leftmost value of the pairs with parallel lines was chosen as $v_{intersect}$ and not the linear interpolated intersection since it belongs to a pair more to the right in the testing direction (u_{min} to u_{max}) from left to right.

If there existed pairs further to the left that have parallel lines, then it was also tested if the backwards generated angles were further away from the finish angle than the two-steps-ahead angles. In this case the leftmost control signal for the pair was used. Figure 3.6 shows an example of the case where no intersection occurs in between any of the pairs.

Before doing any of these tests it was tested if the deadbeat control signal was feasible by using the current state vector and the finish angle as in Sec. 3.5.

3.7 Lock to first approach

The "Lock to first" approach was based on storing data from the first intersection between the forward and backward paths and use this in later samples instead of calculating new intersections at each sample.

Similarly to the "Temporal interpolation" approach in Sec. 3.4 a backwards path was generated for both the current state vector and the next state vector, see Fig. 3.7. The intersection was considered to occur when the next angle θ_n was over the next backwards generated angle θ_{nb}. In this case the two backward paths were stored as two lists of state vectors for each time-sample from the current sample till the sample when the paths reach the finish. The next backward path was also sampled one time step further back than its intersection point and called θ_{pb}.

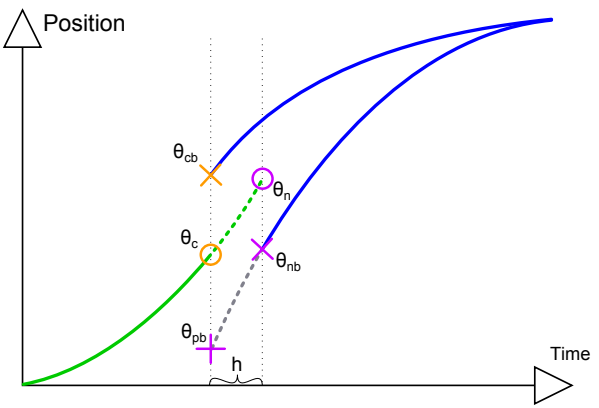

Figure 3.7 Here the samples of the different paths used for the "Lock to first" approach can be seen. θ_c is the current angle, θ_{cb} is the corresponding backwards generated angle. θ_n and $\theta_{nb}$are the same as the above but for the state vector one time step ahead with u_{max} as control signal. θ_{pb} is the sample one time step further back of the backward generated for the same path.

Based on the angles θ_{pb}, θ_{cb} and θ_c an interpolation was done such that the interpolation of the two paths yielded exactly θ_c at the current time step, see Fig. 3.8. This interpolation constant was then saved and used to interpolate each state vector in the two lists to merge them into one path. This path was then appended to the list of state vectors saved from the forward path to generate a complete trajectory.

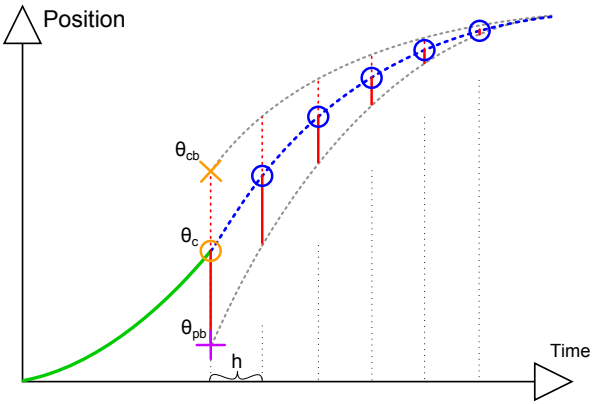

Figure 3.8 Here the interpolation used for "Lock to first" approach can be seen. The solid red lines represent the interpolation between the samples of the two backward generated paths, see Fig. 3.7. The blue circles represent the final path after interpolating.

For this approach it proved important that the forward and backward paths reached the same maximum velocity. The method used previously for handling the maximum velocity in the forward and backward stepping functions yielded varying limit values of the velocity. Therefore an exacter implementation of the forward and backward stepping functions were created and used. First the next state vector was calculated given the current input signal. Then the time t necessary to go from the yielded torque to zero acceleration was calculated. If the velocity after the time t exceeded the maximum velocity ω_{max} then the current input signal could not be applied. The time t was then rounded down to the nearest sample time and used as the constant k in equation system Eq. 3.12.

$$\begin{aligned} x_1 &= A\,x_0 + B\,u_1 \\ x_k &= A(k-1)\,x_1 + B(k-1)\,u_{min} \\ x_{k+1} &= A\,x_k + B\,u_2 \\ x_{k+2} &= A\,x_{k+1} + B\,u_{min} \end{aligned} \tag{3.12}$$

The state vector x_{k+2} is the state vector with the velocity ω_{max} and zero acceleration; x_0 is the current state vector; A and B are the discrete-time state-space matrices for the system and $A(k-1)$ and $B(k-1)$ are matrix functions representing

the zero-order-hold sampled system with sample time $(k-1)\cdot h$. Solving for u_1 and u_2 yields the control sequence

$$\left(u_1,\ u_{min},\ \ldots,\ u_{min},\ u_2,\ u_{min}\right) \tag{3.13}$$

which takes the system from the current state vector to the maximum velocity without exceeding the limits.

Just as a test to see if the complete trajectory generated by this approach actually was possible to follow, given the initial limitations on the system, the generated trajectories were also sent to a deadbeat controller and simulated in forward time.

3.8 Time-Optimal Comparison

Since the backward and forward models used a constant inertia and external torque the corresponding maximal acceleration would also be constant. This means that the trajectory generation using the worst case scenario, discussed in the introduction, will be the real optimal-time solution. It was therefore chosen that trajectories generated for fixed jerk, acceleration and velocity would be used for benchmarking the different approaches. The Reflexxes motion library [11] was used for generating the trajectories.

The Reflexxes motion library is a library for on-line trajectory generation [11]. It is used for trajectory planning as discussed in the introduction but with the benefit of being very fast and light weight such that it can be used in real time. It works with fixed limits on velocity, acceleration and jerk trough out the whole trajectory and must therefore work with the worst case limits to generate feasible trajectories.

The example program "01_RMLPositionSampleApplication" in Type IV RML 1.3.2 Academic version of Kroeger's Reflexxes library [12] was modified to only handle one degree of freedom. The unit of the position was chosen to be *rad*. The acceleration a and jerk $\dot{a}$ limit was scaled according to

$$\begin{aligned} I_{inertia}a(t) = \tau(t),\ \ \dot{\tau} = u(t) \\ \Downarrow \\ a(t) = \frac{\tau(t)}{I_{inertia}},\ \ \dot{a}(t) = \frac{u(t)}{I_{inertia}} \end{aligned} \tag{3.14}$$

and the generated acceleration profile return by Reflexxes was then scaled back to torque to make for an easy comparison against the generated torque from the approaches tested.

4

Results

4.1 Lower arm moment of inertia

Parallel arm

See Sec. 2.4 for theory on calculations and Sec. 3.1 for experiment setup.

	T [s]	N [1]	ω_e [rad/s]
Test 1	164.62	100	3.818
Test 2	329.91	200	3.809

ω_e was chosen to

$$\omega_e \approx 3.81 \text{ [rad/s]} \tag{4.1}$$

and the following was measured

$$\begin{aligned} m &= 0.085 \text{ [kg]} \\ l_{mass} &= 0.372 \text{ [m]} \end{aligned} \tag{4.2}$$

This gives (with $g = 9.81$ [m/s^2])

$$I_{centre} \approx 0.00961 \text{ [kg m}^2\text{]} \tag{4.3}$$

around centre of mass in the x- and z-directions.

4.2 Middle arm moment of inertia

Top arm

See Sec. 2.4 for theory on calculations and Sec. 3.1 for experiment setup.

	T [s]	N [1]	ω_e [rad/s]
Test 1	134.59	110	5.135
Test 2	134.73	110	5.130

ω_e was chosen to

$$\omega_e \approx 5.13 \text{ [rad/s]} \tag{4.4}$$

and the following was measured

$$\begin{aligned} m &= 0.575 \text{ [kg]} \\ l_{mass} &= 0.286 \text{ [m]} \end{aligned} \tag{4.5}$$

This gives (with $g = 9.81$ [m/s^2])

$$I_{centre} \approx 0.0141 \text{ [kg m}^2\text{]} \tag{4.6}$$

around centre of mass in the x- and z-directions.

Lower arm

See Sec. 2.4 for theory on calculations and Sec. 3.1 for experiment setup.

	T [s]	N [1]	ω_e [rad/s]
Test 1	82.84	60	4.551
Test 2	137.65	100	4.565

ω_e was chosen to

$$\omega_e \approx 4.56 \text{ [rad/s]} \tag{4.7}$$

and the following was measured

$$\begin{aligned} m &= 0.22 \text{ [kg]} \\ l_{mass} &= 0.3695 \text{ [m]} \end{aligned} \tag{4.8}$$

This gives (with $g = 9.81$ [m/s^2])

$$I_{centre} \approx 0.00843178430 \text{ [kg m}^2\text{]} \tag{4.9}$$

around center of mass in the x- and z-directions.

4.3 Upper arm moment of inertia

See Sec. 2.4 under “From oscillation torque” for theory on calculations and Sec. 3.1 for experiment setup.

The torque τ is the torque measured from the axis servos where friction and gravitation compensation have been included.

A [rad]	ω [rad/s]	$E(I)$ [kg m^2]	$V(I)$
0.038	94.25 (15 Hz)	0.2111	0.0184
0.038	62.83 (10 Hz)	0.2199	0.0126
0.038	31.42 (5 Hz)	0.2039	0.033

The final moment of inertia was chosen to $I = 0.22$ kg m^2 since the 10 [Hz] result has the lowest variance and the 15 [Hz] estimate is close to that. The 5 Hz result was ignored since its variance is much higher. The reason it is higher is probably that the torque is lower in comparison to the friction. All of the frequencies have high variance thought, and this might come from the phase shift between the position reference and the actual position. Better results might be acquired if the actual acceleration was used for calculating the inertia instead of using the analytical acceleration of sinusoidal reference.

4.4 Friction

Figure 4.1 shows the friction torque for axis 1 at two different speeds. The blue graph shows the friction at 2.18 deg/s and the yellow at 43.64 deg/s. The values represent the mean torque over 0.109 degree intervals. The standard deviation for each interval is shown as the region with the same colour around the mean values. The torque is measured at the axis servo so the torque equivalent on the upper arm is 33 times higher than the values shown because of the gearing.

By feedforwarding the required torque for the low speed and then running at the two speeds again, the friction effect could be reduced substantially, see Fig. 4.2, not just for the low speed but also for the higher.

If the torque disturbance that is left for the higher speed is added to the feedforward, see magenta graph in Fig. 4.3, the disturbance effect is the same as when using just the lower speed values. So only the Coulomb frictions position dependence seems to be possible to compensate for, not the viscous speed-dependent part. For the speed depending friction mean values from Rosenquist's thesis [13] was used instead.

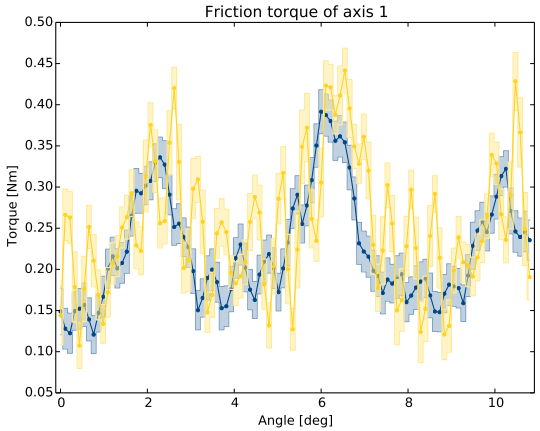

Figure 4.1 Here the mean friction torque, for 0.109 degree intervals, for two different speeds can be seen. The blue graph shows the friction at 2.18 deg/s and the yellow at 43.64 deg/s. The standard deviation for each interval is shown as coloured regions around the mean values.

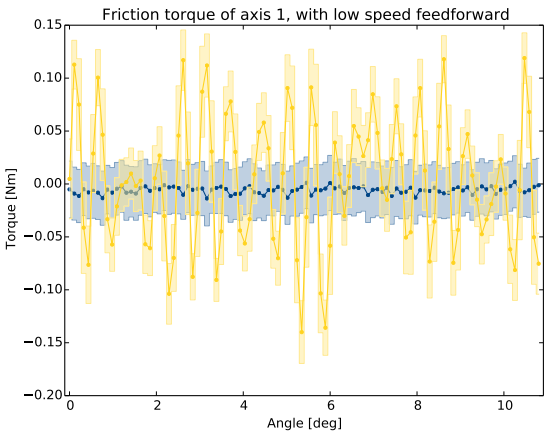

Figure 4.2 Here the results of the same test as in Fig. 4.1 can be seen, but with the mean values for the low speed used as torque feedforward.

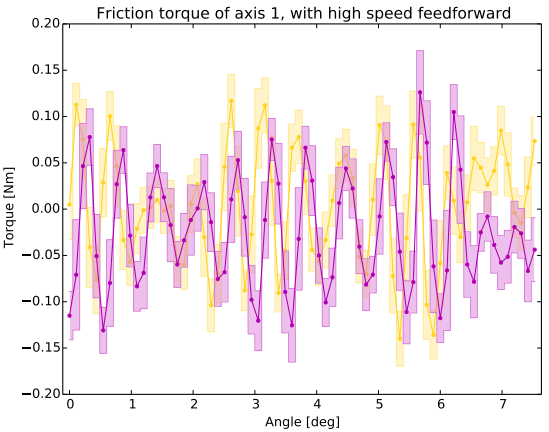

Figure 4.3 Here the mean torque (for each 0.109 degree interval) for the higher speed with different torque feedforward can be seen. The yellow graph is with feedforward values from the low speed as in Fig. 4.2 and the magenta graph is with values from the high speed test in Fig. 4.1.

4.5 Model of one axis

Least squares results

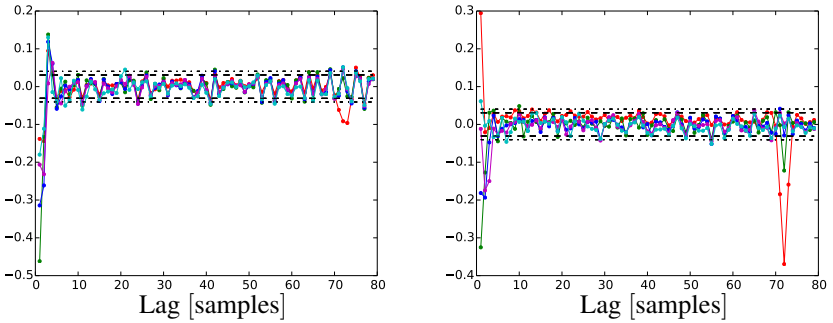

Figure 4.4 This figure shows the autocorrelation of the residual for the least-squares models. The left diagram is for validation with 0.2 Nm torque amplitude and the right with 0.5 Nm. Red curves are for model order $n = 1$, green are for order $n = 2$, blue are for order $n = 3$, magenta are for order $n = 4$ and cyan are for order $n = 30$. Black dashed lines are limits for 95 % rejection certainty for white noise residuals and point dashed lines are limits for 99 %.

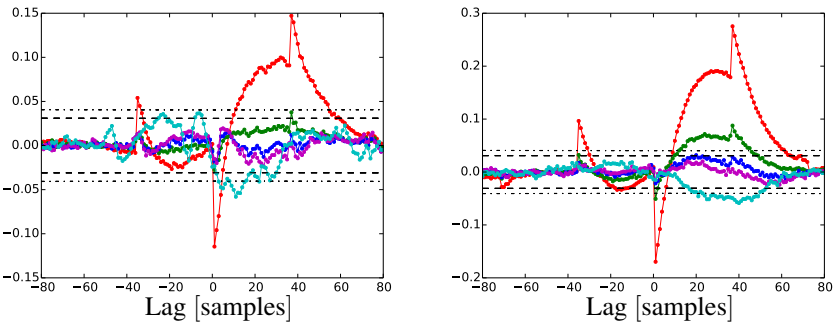

Figure 4.5 This figure shows the cross correlation between the residual and the input signal for the least-squares models. The left diagram is for validation with 0.2 Nm torque amplitude and the right with 0.5 Nm. Red curves are for model order $n = 1$, green are for order $n = 2$, blue are for order $n = 3$, magenta are for order $n = 4$ and cyan are for order $n = 30$. Black dashed lines are limits for 95 % rejection certainty for white noise residuals and point dashed lines are limits for 99 %.

Order	τ_{amp} [Nm]	σ of residual	max of parameter covariance matrix
1	0.2	0.03619	1.027e-06
1	0.5	0.05113	2.050e-06
2	0.2	0.03613	0.0001261
2	0.5	0.03503	0.0001185
3	0.2	0.03005	0.0003334
3	0.5	0.02755	0.0002801
4	0.2	0.02749	0.0003070
4	0.5	0.02599	0.0002746
30	0.2	0.02606	0.0003501
30	0.5	0.02517	0.0003268

Table 4.1 This table shows the standard deviation σ of the residual and the maximum value of the covariance matrix of the estimated parameters for different model order. tau_{amp} is the PRBS signal amplitude used for model validation.

Order	a	b
1	$\begin{pmatrix}-0.9319\end{pmatrix}$	$\begin{pmatrix}0.05993\end{pmatrix}$
2	$\begin{pmatrix}0.2475\\-1.2169\end{pmatrix}$	$\begin{pmatrix}-0.1568\\0.1856\end{pmatrix}$
3	$\begin{pmatrix}-0.01252\\-0.1184\\-0.8323\end{pmatrix}$	$\begin{pmatrix}-0.1342\\-0.01394\\0.1845\end{pmatrix}$
4	$\begin{pmatrix}0.006465\\-0.1690\\-0.1823\\-0.6089\end{pmatrix}$	$\begin{pmatrix}-0.07216\\-0.09302\\0.02745\\0.1840\end{pmatrix}$

Table 4.2 This table shows the estimated model parameters for the different model orders.

Instrumental variable method results

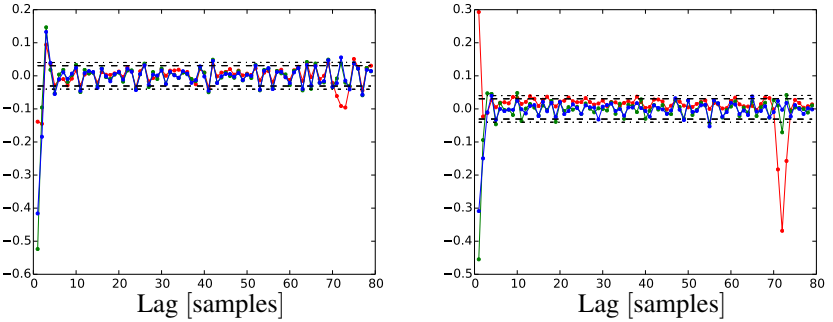

Figure 4.6 This figure shows the autocorrelation of the residual for the instrumental variable methods models. The left diagram is for validation with 0.2 Nm torque amplitude and the right with 0.5 Nm. Red curves are for model order $n = 1$, green are for order $n = 2$ and blue are for order $n = 3$. Black dashed lines are limits for 95 % rejection certainty for white noise residuals and point dashed lines are limits for 99 %.

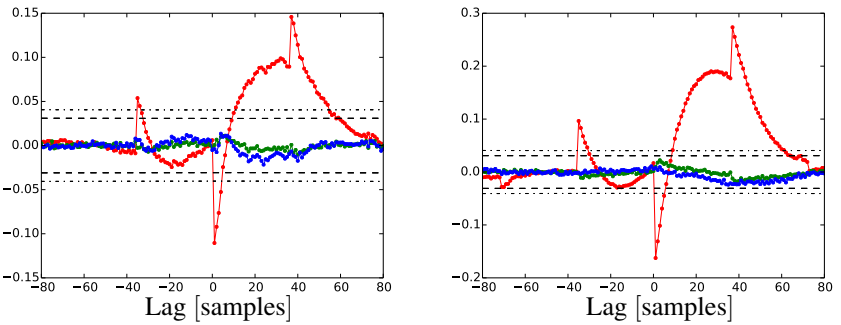

Figure 4.7 This figure shows the cross correlation of the residual and the input signal for the instrumental variable methods models. The left diagram is for validation with 0.2 Nm torque amplitude and the right with 0.5 Nm. Red curves are for model order $n = 1$, green are for order $n = 2$ and blue are for order $n = 3$. Black dashed lines are limits for 95 % rejection certainty for white noise residuals and point dashed lines are limits for 99 %.

Order	τ_{amp} [Nm]	σ of residual	max of parameter covariance matrix
1	0.2	0.03620	1.002e-06
1	0.5	0.05114	1.999e-06
2	0.2	0.04022	0.0002097
2	0.5	0.03637	0.0001715
3	0.2	0.03211	0.002223
3	0.5	0.02803	0.001694

Table 4.3 This table shows the standard deviation σ of the residual and the maximum value of the covariance matrix of the estimated parameters for different model order. tau_{amp} is the PRBS signal amplitude used for model validation.

Order	a	b
1	$\begin{pmatrix} -0.9322 \end{pmatrix}$	$\begin{pmatrix} 0.05951 \end{pmatrix}$
2	$\begin{pmatrix} 0.4596 \\ -1.4453 \end{pmatrix}$	$\begin{pmatrix} -0.1770 \\ 0.1915 \end{pmatrix}$
3	$\begin{pmatrix} -0.01097 \\ 0.06452 \\ -1.0266 \end{pmatrix}$	$\begin{pmatrix} -0.1029 \\ -0.05966 \\ 0.1900 \end{pmatrix}$

Table 4.4 This table shows the estimated model parameters for the different model orders.

As can be seen in Figs. 4.4 - 4.8 the models behave very similarly for the 0.2 Nm and 0.5 Nm amplitudes which means that the system can be considered linear up to at least 0.5 Nm. Due to problems with too large movements, higher torques were not tested. But the system was assumed to be linear for higher torques as well.

None of the model orders for the least-squares estimates removed the problems with the autocorrelation. This implies that the residuals are not white noise and their estimates may be biased. Instrumental variable method estimates have only slightly higher maximal parameter covariance than for the least-squares estimation so there should be sufficient correlation between the instrumental variable and the measurements, see Sec. 2.5. As can be seen in Fig. 4.7, the first-order estimate can easily be rejected. The second and third however both stay within the 95 % rejection limits so they cannot be rejected.

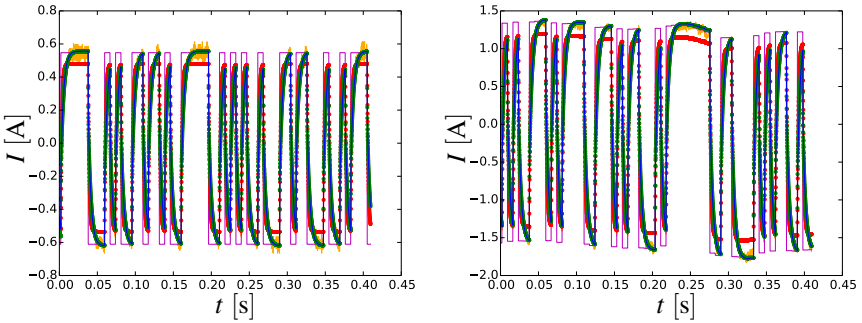

Figure 4.8 This figure shows the step response of the instrumental variable method estimates. The left is for 0.2 Nm torque amplitude and the right is for 0.5 Nm. Red curves are for model order $n = 1$, green are for order $n = 2$ and blue are for order $n = 3$. The magenta curves are the input reference and orange are the actual current.

Looking at Fig. 4.8 it is obvious that both the second-order and third-order estimates are very similar. Because of this the second-order estimate was chosen. This model has the discrete-time transfer function

$$H(z) = \frac{-0.1770z + 0.1915}{z^2 + 0.4596z - 1.4453} \tag{4.10}$$

and the equivalent zero-order hold continuous-time transfer function is

$$\begin{aligned} G(s) &= \frac{2752s + 2.100 \cdot 10^6}{s^2 + 7667s + 2.099 \cdot 10^6} \\ &= 2752 \frac{s + 763.2}{(s + 284.3)(s + 7383)} \end{aligned} \tag{4.11}$$

The poles of this second order estimate are placed at -284.3 and -7383 which means that the current loop has very fast dynamics. According to Sec. 2.7 the poles for the backward time system change sign and therefore the backwards model will become very unstable. Only one limit is known for the current loop, the minimal and maximal torque, so only one degree of freedom can be limited with this. Therefore one mode of the model may increase uncontrollably and this will lead to numerical problems. This could be fixed by adding a limit on the time derivative of the torque, but one problem remains. The current loop has two poles and therefore will contribute with two states in the state-space model. Together with the angle and velocity states this yields a total of four states. This makes the intersection between forward and backward paths much more complicated since four states have to be equal, which means that two synchronizing control signals has to be used.

4.6 Time-Optimal Comparison

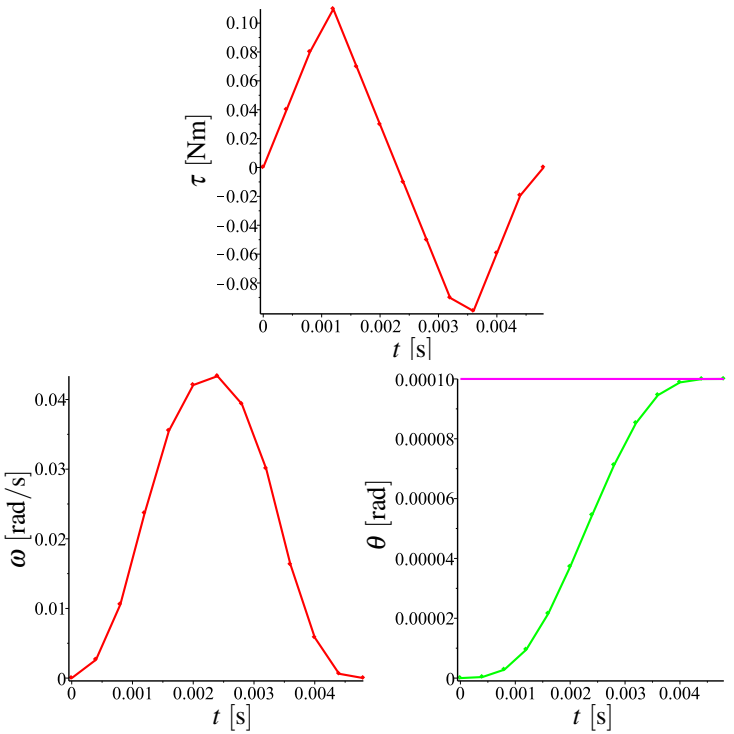

Figure 4.9 This is the output from the Reflexxes library for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top diagram is the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. The control sequence ends at $t = 0.0048$ s.

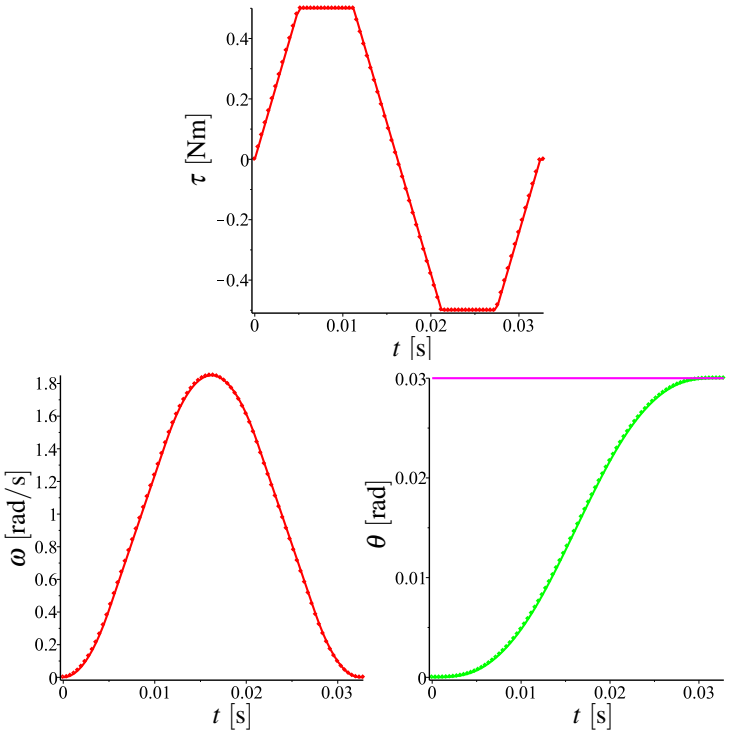

Figure 4.10 This is the output from the Reflexxes library for a step trajectory input. The step height is 0.03 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top diagram is the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle and the purple graph shows the proposed trajectory. The control sequence ends at $t = 0.0328$ s.

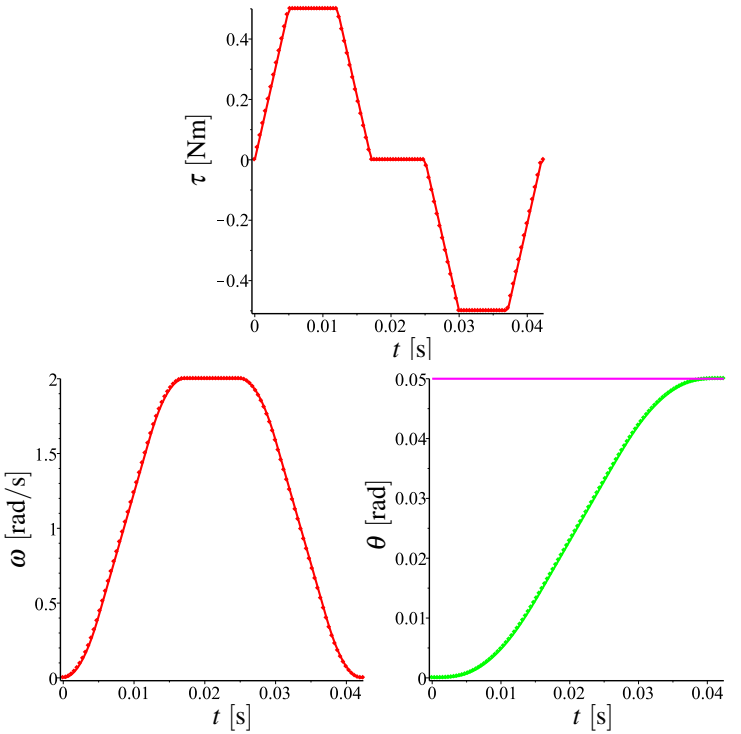

Figure 4.11 This is the output from the Reflexxes library for a step trajectory input. The step height is 0.05 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top diagram is the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. The control sequence ends at $t = 0.0424$ s.

4.7 Temporal interpolation approach

When setting the $t_{sample\ ahead}$ parameter for the "Temporal interpolation" approach to zero the control sequence becomes very jumpy, as can be seen in Fig. 4.12, which is an unwanted effect that would increase wear on a physical system. The problem here is that the intersection point is sent to the deadbeat as reference. Since the system is of third order it will take the deadbeat three sample times to reach the reference, but this is only true when the reference is stationary. In the case that the reference for example is a ramp, there will always be an error left. In order for the control signal to be optimal the system has to be able to follow the backwards

generated path exactly once an intersection has occurred, but since the backward path is not stationary this will not be possible.

The biggest problem with the error for this approach is that the intersection point is recalculated in each time step. So the next reference sent to the deadbeat controller will not be a new sample of the first intersection path, but the new intersection point with the current state, which already has an error compared to the optimal case (the first backwards intersection path). This means that the error will accumulate with each time step.

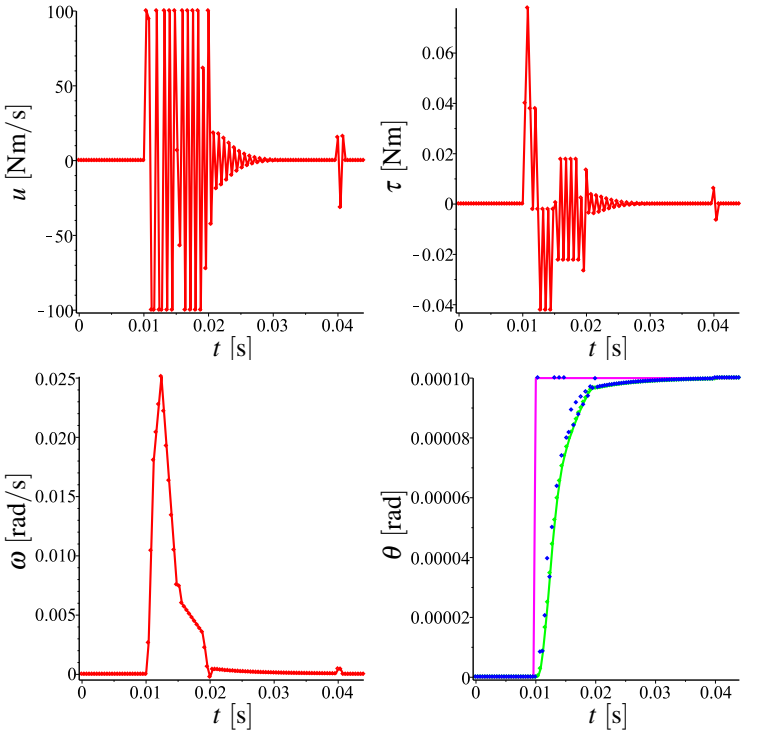

Figure 4.12 This is the output from the "Temporal interpolation" approach for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 1$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, $t_{sample\ ahead} = 0h$ and $h = 0.4$ ms. The top left diagram shows the control signal generated by the deadbeat controller. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory and the blue-dotted graph show what positions where sent to the deadbeat controller. The controller is inactive after $t = 0.0408$ s and the step occurs at $t = 0.01$ s.

The idea behind the $t_{\text{sample ahead}}$ parameter was to compensate for this error by giving the deadbeat the reference earlier so that it can reach it in time. Though, putting this parameter too high will result in an error in the other direction that accumulates and results in over-shooting the final state, see Fig. 4.13.

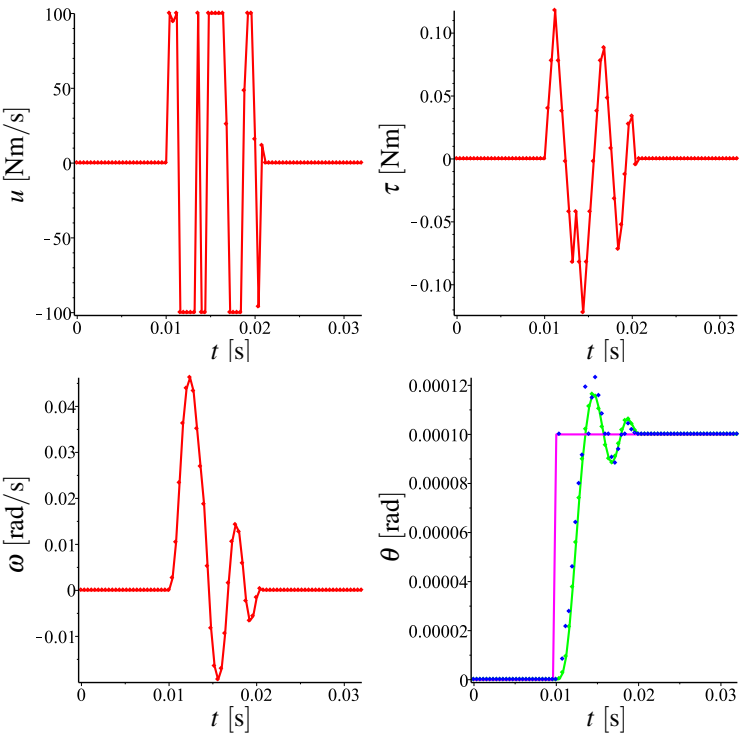

Figure 4.13 This is the output from the "Temporal interpolation" approach for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 1$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, $t_{sample\ ahead} = 2h$ and $h = 0.4$ ms. The top left diagram shows the control signal generated by the deadbeat controller. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory and the blue-dotted graph show what positions where sent to the deadbeat controller. The controller is inactive after $t = 0.0212$ s and the step occurs at $t = 0.01$ s.

Using $t_{\text{sample ahead}} = 1.9h$ shows promising for the 0.0001 rad step, see Fig. 4.14 since the generated control sequence reaches the finish state in 0.0052 s ($0.0152s - 0.01$ s) which is only 8.33 % (equal to one sample time) longer than the time-optimal, see Fig. 4.9.

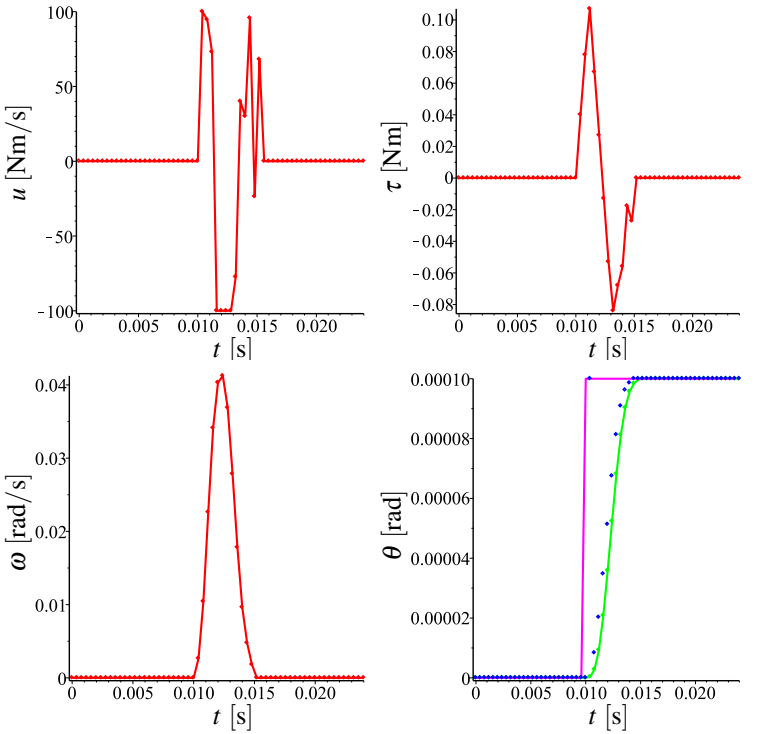

Figure 4.14 This is the output from the "Temporal interpolation" approach for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 1$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, $t_{sample\ ahead} = 1.9h$ and $h = 0.4$ ms. The top left diagram shows the control signal generated by the deadbeat controller. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory and the blue-dotted graph show what positions where sent to the deadbeat controller. The controller is inactive after $t = 0.0152$ s and the step occurs at $t = 0.01$ s.

For the steps where the torque and velocity limits start to play a role the control signal starts to jump between minimum and maximum. The optimality is also much lower as it takes the control sequence 0.0388 s ($0.0488s - 0.01$ s) to reach the final state for the 0.03 rad step, see Fig. 4.15, which is 18.3 % longer compared to the 0.0328 s for the time-optimal, see Fig. 4.10. For the 0.05 rad step the time is up at 0.0644 s, see Fig. 4.16, compared to the time-optimal 0.0424 s (more than 50 % longer), see Fig 4.11.

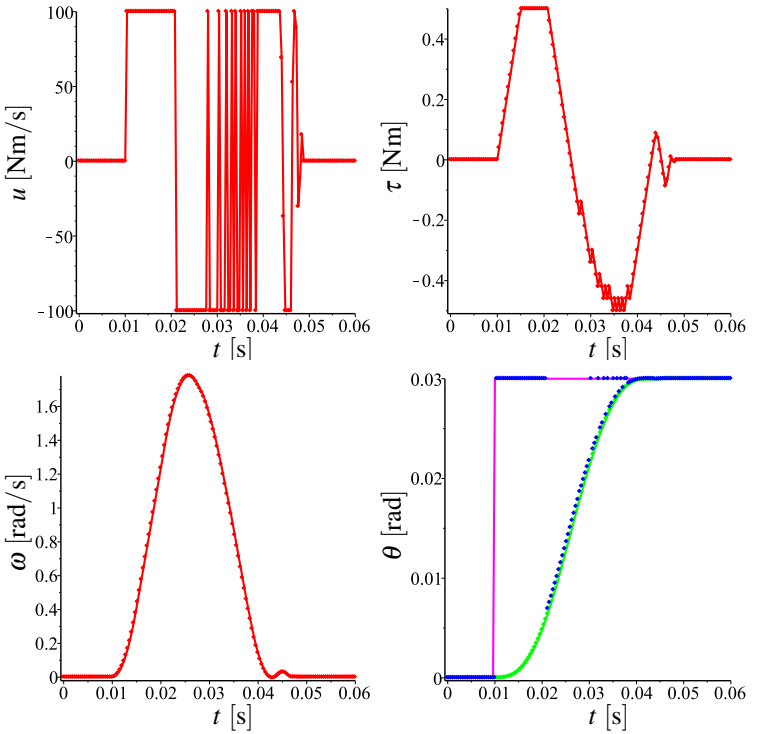

Figure 4.15 This is the output from the "Temporal interpolation" approach for a step trajectory input. The step height is 0.03 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, $t_{sample\ ahead} = 1.9h$ and $h = 0.4$ ms. The top left diagram shows the control signal generated by the deadbeat controller. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory and the blue-dotted graph show what positions where sent to the deadbeat controller. The controller is inactive after $t = 0.0488$ s and the step occurs at $t = 0.01$ s.

This is probably due to the fact that the derivative of the backwards generated path is different depending on the limiting case (since different cases generate different control signals, see Sec. 2.6). So $t_{\text{sample ahead}} = 1.9h$ will only compensate for the error in certain cases and a constant $t_{sample\ ahead}$ therefore seems unlikely to work. It might be that a varying $t_{\text{sample ahead}}$, according to some rule, may fix this. This was not tested for this thesis but it feels like an unreliable attempt to fix the underlying problem with the error.

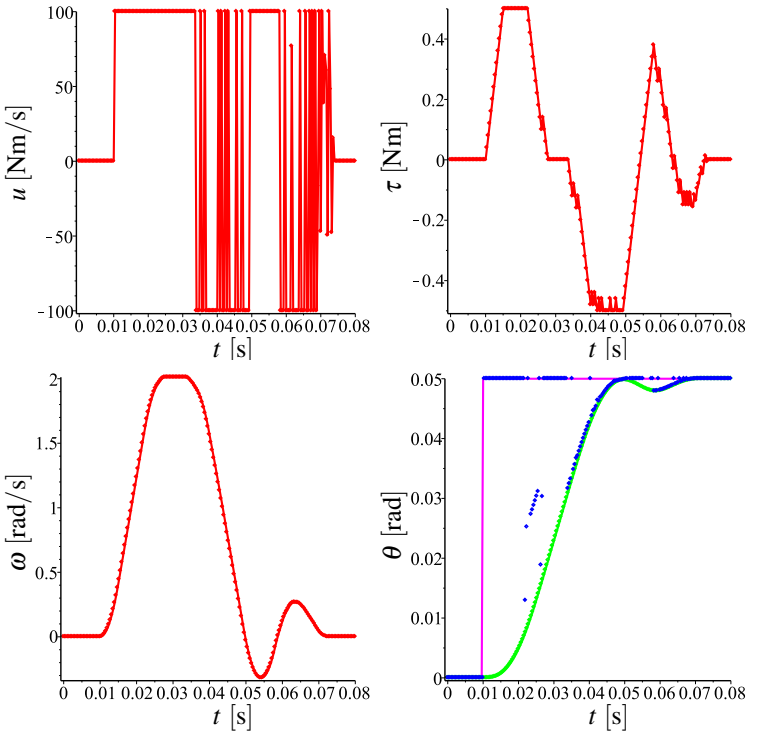

Figure 4.16 This is the output from the "Temporal interpolation" approach for a step trajectory input. The step height is 0.05 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, $t_{sample\ ahead} = 1.9h$ and $h = 0.4$ ms. The top left diagram shows the control signal generated by the deadbeat controller. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory and the blue-dotted graph show what positions were sent to the deadbeat controller. The controller is inactive after $t = 0.0744$ s and the step occurs at $t = 0.01$ s.

4.8 State feedback controlled intersection error

The "State feedback controlled intersection" approach has a big disadvantage already from the start, since the limits on the control signal for the backwards path are stricter than the real limits (90 %). For the two test with different time constants for the 0.0001 rad step it can be seen that the controller with faster time constants, see Fig. 4.17, saturates the control signal. This results in an over-shoot and very

poor speed (0.01 s which is 108 % longer than the optimal-time). The controller with slower time constants yields a much better result, see Fig. 4.18, but is still very slow (0.0068 s which is 42 % longer than the time-optimal).

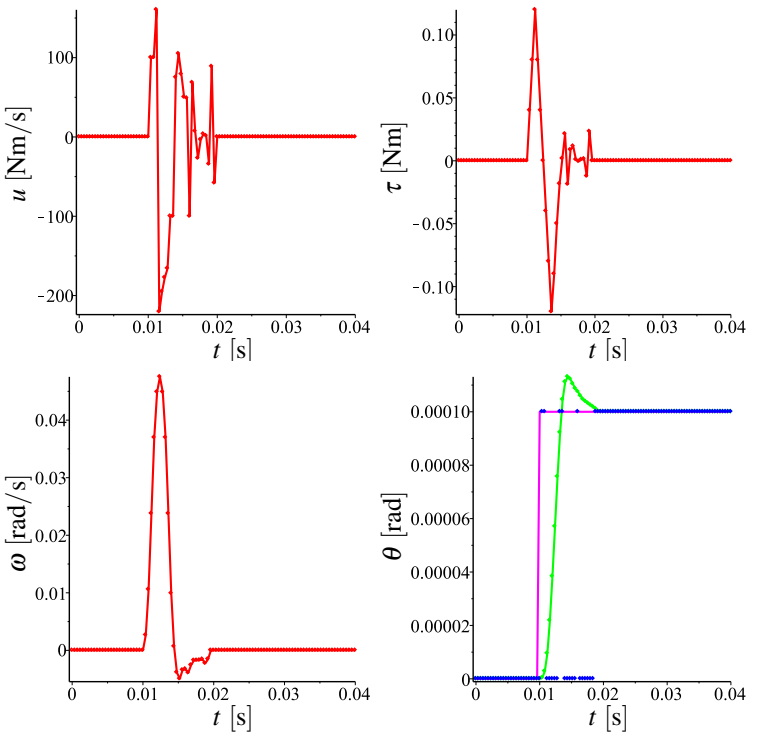

Figure 4.17 This is the output from the "State feedback controlled intersection" approach for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, time constants for the poles $[3h, 3h/2, h]$ and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the state feedback controller is active for that sample time. The controller is inactive after $t = 0.0200$ s and the step occurs at $t = 0.01$ s.

For the smaller step trajectory at 0.00002 rad the faster controller does not saturate the control signal and yields result without over-shoot, but the time is still poor since it still is slower than the time-optimal is for a five times larger step. The slower controller is in this case slower than the faster controller and reaches the final state

after almost the same time as it did for the 0.0001 rad step. This is due to the fact that the time constants determine how much time it will take to remove the error. This means that no controller will work well for all step sizes. It might be possible to change the time constants dynamically but this will become very complicated and the stability of the controller may be hard to guarantee, so this is not a recommended approach.

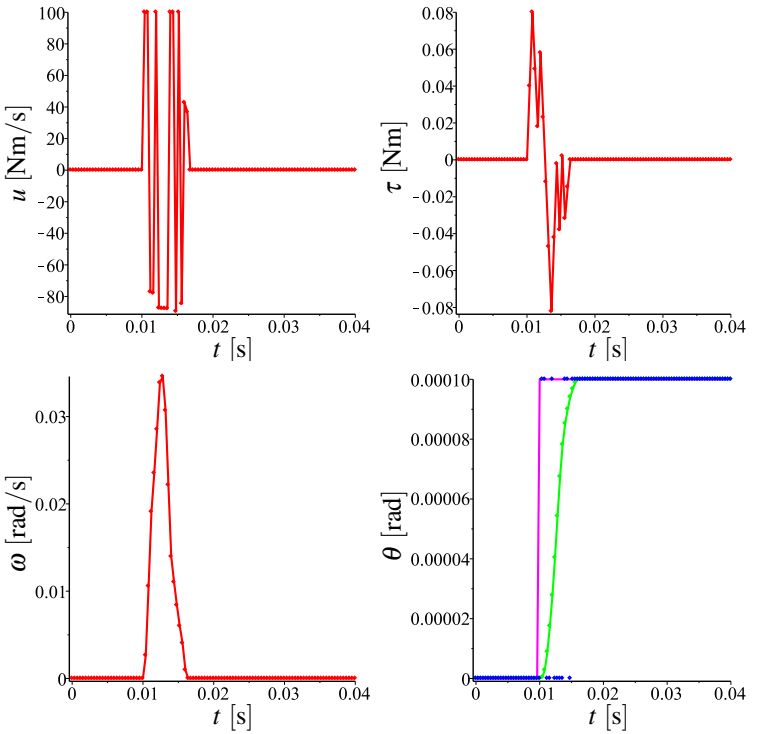

Figure 4.18 This is the output from the "State feedback controlled intersection" approach for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, time constants for the poles $[10h, 5h, 10h/3]$ and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the state feedback controller is active for that sample time. The controller is inactive after $t = 0.0168$ s and the step occurs at $t = 0.01$ s.

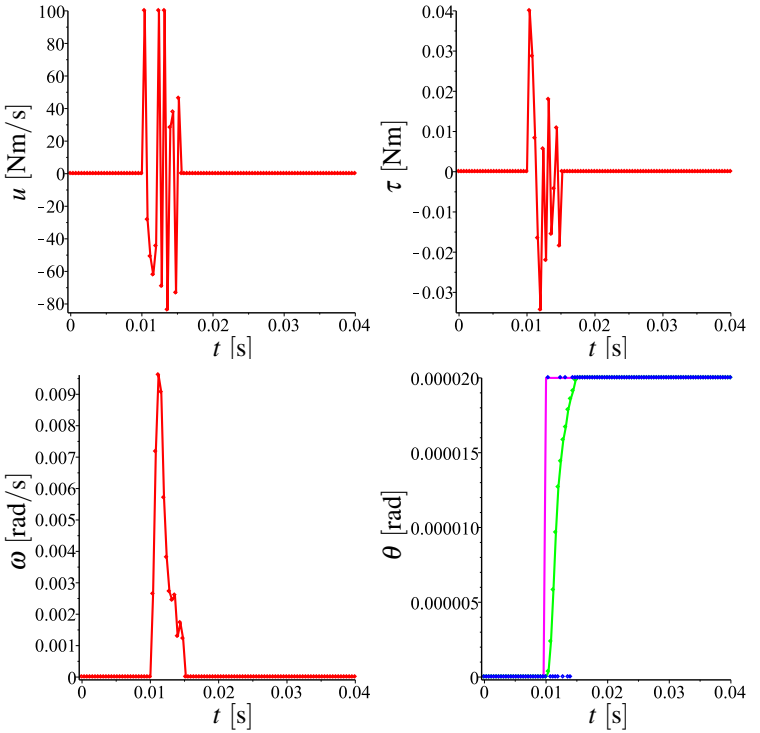

Figure 4.19 This is the output from the "State feedback controlled intersection" approach for a step trajectory input. The step height is 0.00002 rad, $u_{max} = -u_{min} =$ 100 Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, time constants for the poles $[3h, 3h/2, h]$ and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the state feedback controller is active for that sample time. The controller is inactive after $t = 0.0156$ s and the step occurs at $t = 0.01$ s.

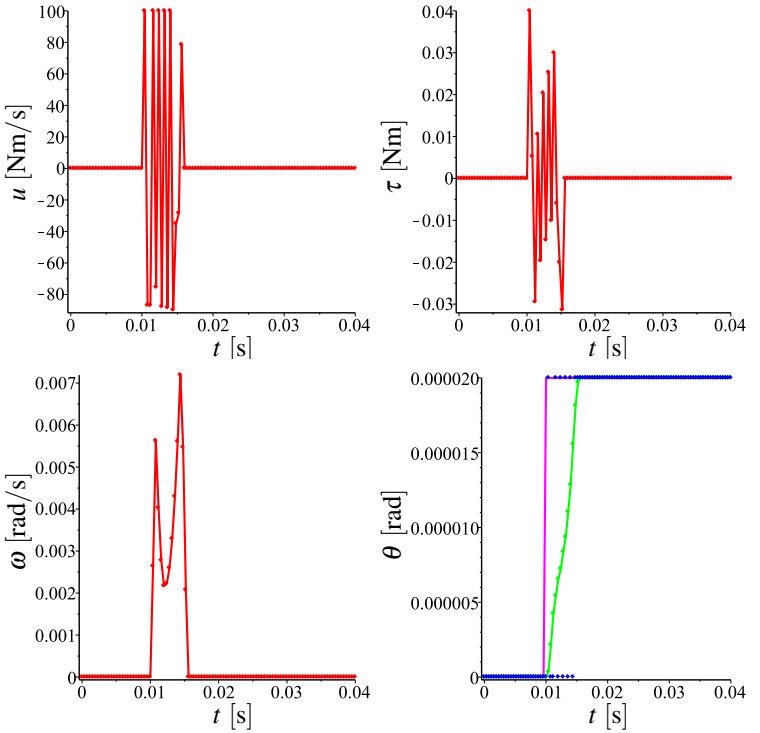

Figure 4.20 This is the output from the "State feedback controlled intersection" approach for a step trajectory input. The step height is 0.00002 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s, time constants for the poles $[10h, 5h, 10h/3]$ and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the state feedback controller is active for that sample time. The controller is inactive after $t = 0.016$ s and the step occurs at $t = 0.01$ s.

4.9 Intersection by control signal

Looking at the results for both the control sequence types for the "Intersection by control signal", see Fig. 4.21 to 4.23 and 4.24 to 4.26, it is clear that this approach is much better at generating smooth control signals than the "Temporal interpolation" and the "State feedback controlled intersection" approaches.

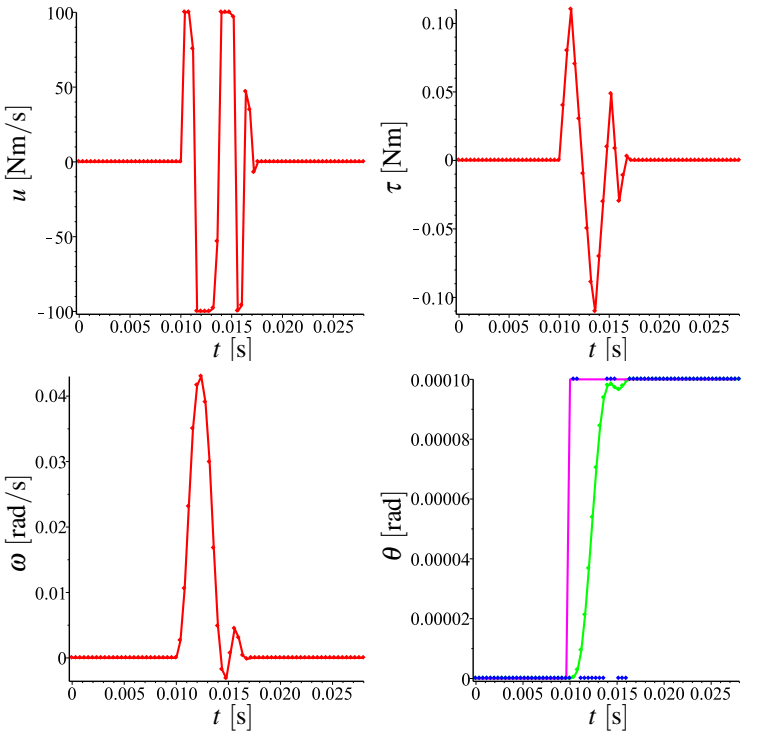

Figure 4.21 This is the output from the "Intersection by control signal" approach, with control sequence of type 1, for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the intersection control signal is used for that sample time. The controller is inactive after $t = 0.0176$ s and the step occurs at $t = 0.01$ s

The “Intersection by control signal” approach is just as the “Temporal interpolation” and the “State feedback controlled intersection” approaches based on recalculating the intersection data at each time step. The benefit of doing this is that the generated trajectory is a result from a simulation with the desired limitations. This means that the trajectory always will be feasible. Any intersection errors that occur because of the limits can then be handled in the next time step. However this also means that the approaches will have closed-loop dynamics which can become very complicated to analyse.

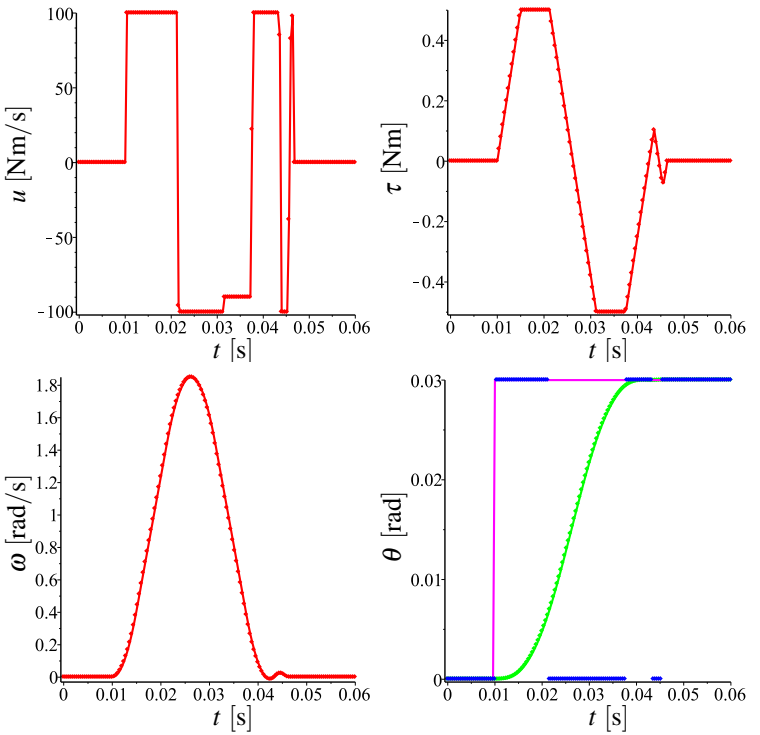

Figure 4.22 This is the output from the “Intersection by control signal” approach, with control sequence of type 1, for a step trajectory input. The step height is 0.03 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the intersection control signal is used for that sample time. The controller is inactive after $t = 0.0468$ s and the step occurs at $t = 0.01$ s

Type 1 control sequence

Theoretically the type-1 control sequence should be able to find the exact solution but somehow the closed loop dynamics together with errors from the linear interpolation in the "maxBrakTest" function, see Sec. 3.2, and other approximations results in oscillations at the end.

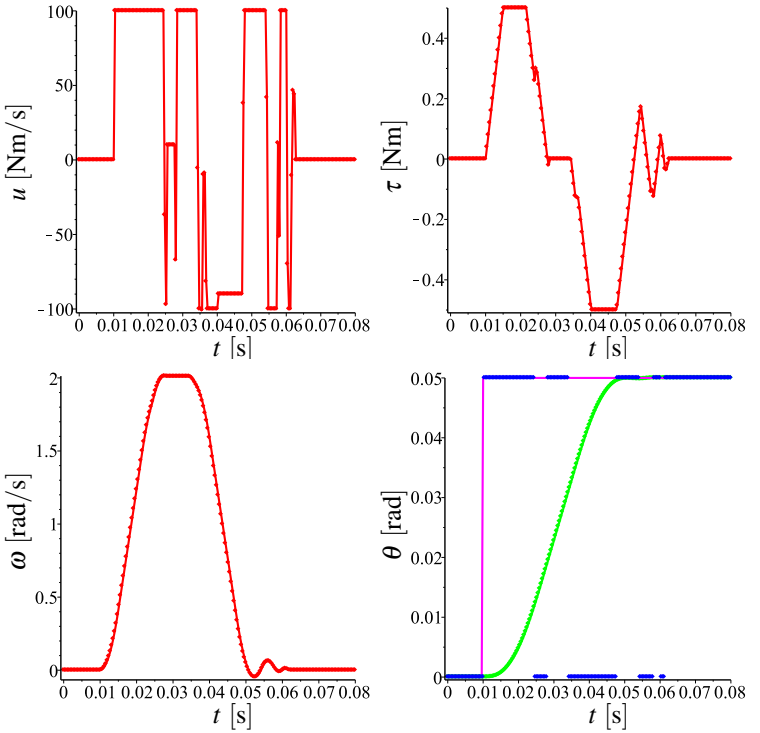

Figure 4.23 This is the output from the "Intersection by control signal" approach, with control sequence of type 1, for a step trajectory input. The step height is 0.05 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the intersection control signal is used for that sample time. The controller is inactive after $t = 0.0628$ s and the step occurs at $t = 0.01$ s.

Due to the oscillations, the time-optimality of the type-1 control sequence is quite poor, with 0.0076 s for the 0.0001 rad step, see Fig. 4.21. Compared to the 0.0048 s for the time-optimal this is 58 % longer. For the higher steps the results are

slightly better, 12 % longer for the 0.03 rad step (see Fig. 4.22) and 24 % for the 0.05 rad (see Fig. 4.23), but still not good.

Type 2 control sequence

The type-2 control sequence is two samples long and therefore has a longer time for the first value to affect the system which generates less aggressive control signals.

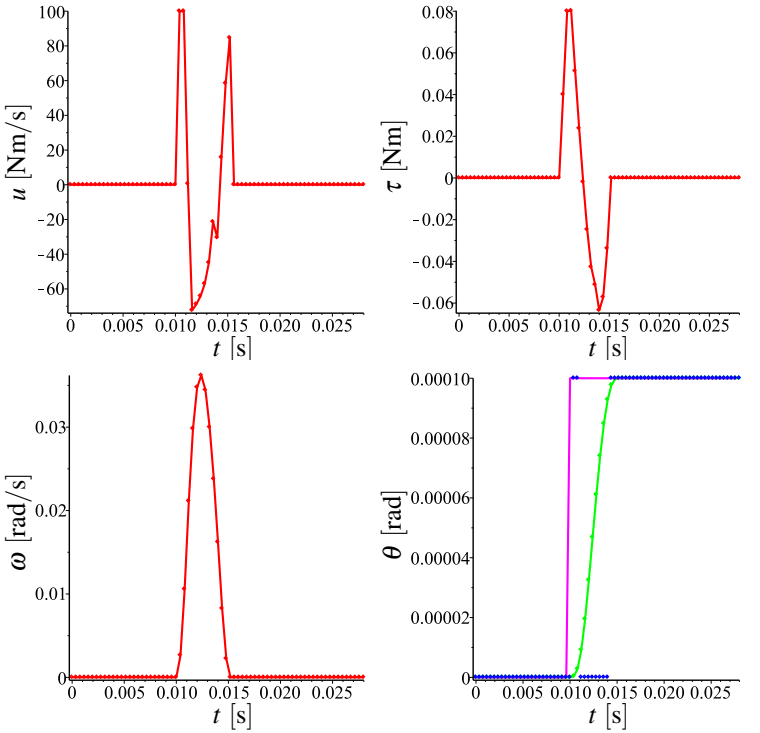

Figure 4.24 This is the output from the "Intersection by control signal" approach, with control sequence of type 2, for a step trajectory input. The step height is 0.0001 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the intersection control signal is used for that sample time. The controller is inactive after $t = 0.0156$ s and the step occurs at $t = 0.01$ s

Since the second part of the control sequence never is used and instead a new sequence is generated at each time step there is more room for the algorithm to

correct for any errors in the previous backwards intersection.

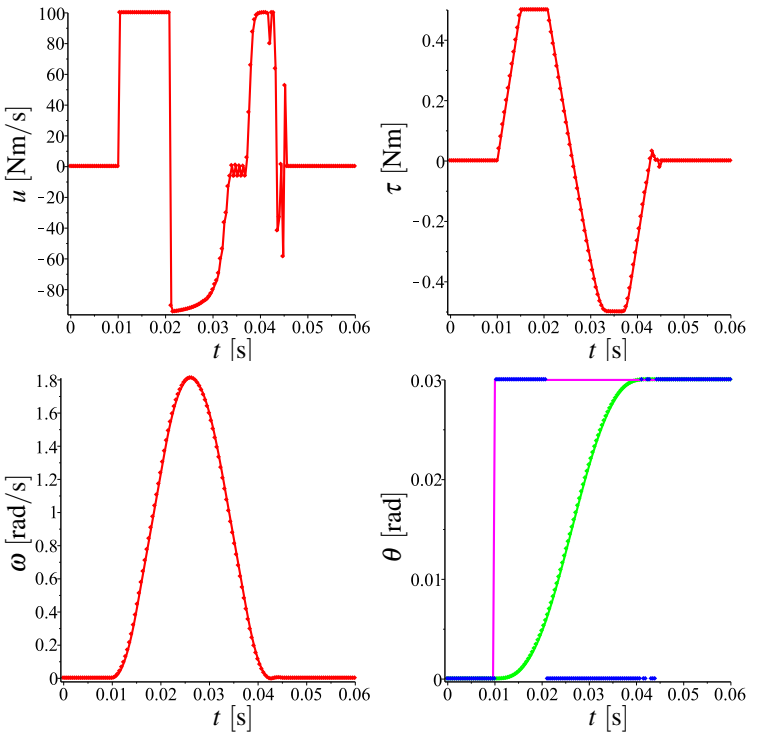

Figure 4.25 This is the output from the "Intersection by control signal" approach, with control sequence of type 2, for a step trajectory input. The step height is 0.03 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the intersection control signal is used for that sample time. The controller is inactive after $t = 0.0456$ s and the step occurs at $t = 0.01$ s

This approach result in much smaller oscillations at the end compared to the type 1 control sequence. So even though this approach will generate less aggressive control signals and therefore be less optimal the fact that it also suppresses the oscillations makes it faster than the type-1. Compared with the time-optimal solution this approach takes 8.5 % longer for the 0.03 rad step (see Fig. 4.25) and 7.5 % for the 0.05 rad step (see Fig. 4.26), some of the improvement for the last step may come from the fact that the method for limiting the velocity allows a few percent

higher velocities than the given limit. During this extra time the angular difference compared to the time-optimal never exceed 0.01% of the step size for the 0.03 rad and the 0.05 rad steps. For the 0.0001 rad step the effectiveness is lower (see Fig. 4.24) with 17 % longer than the time-optimal and angular difference smaller than 0.3% compared to the time-optimal.

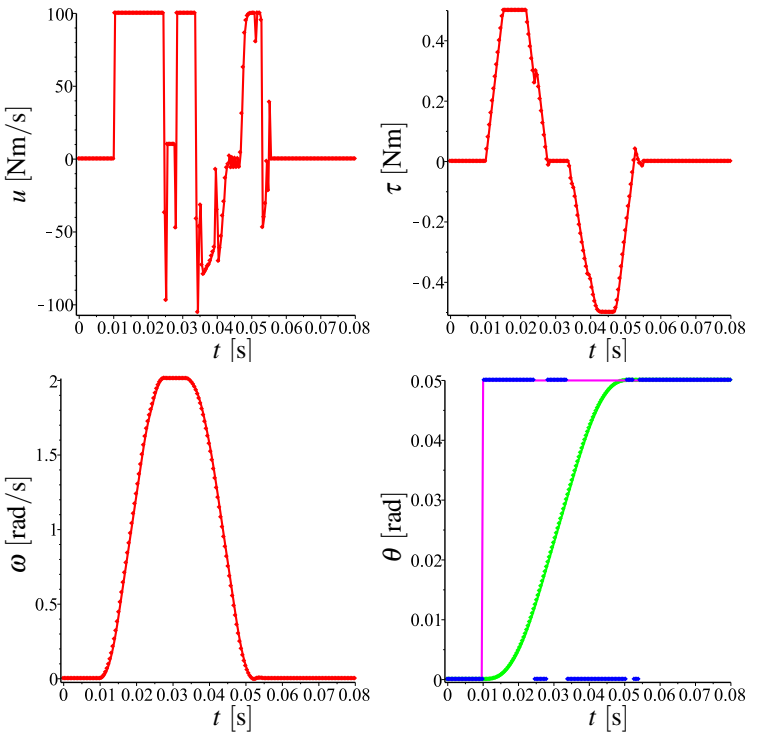

Figure 4.26 This is the output from the "Intersection by control signal" approach, with control sequence of type 2, for a step trajectory input. The step height is 0.05 rad, $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal sent to the forward iterating function. The top right diagram represents the torque and the bottom left represents the velocity. The green curve in the bottom right diagram represents the angle; the purple graph shows the proposed trajectory. If the blue-dotted graph has the value zero that means that the intersection control signal is used for that sample time. The controller is inactive after $t = 0.0556$ s and the step occurs at $t = 0.01$ s.

4.10 Lock to first approach

In contrast to the other approaches in this thesis the "Lock to first" approach has no closed-loop dynamics and is fully based on patching together the forward with the backward path at the intersection point. The patching time-step does not take any of the limits on the system into account. This means that the resulting trajectory may not be possible to follow due to jumps of the states at the patching point. However, as can be seen in the following results this patching time step is no problem with the interpolation described in Sec. 3.7. The benefits of the "Lock to first" approach is that no errors from approximations can accumulate because of feedback and it also consumes less computational power since the intersection does not have to be recalculated in each time-step.

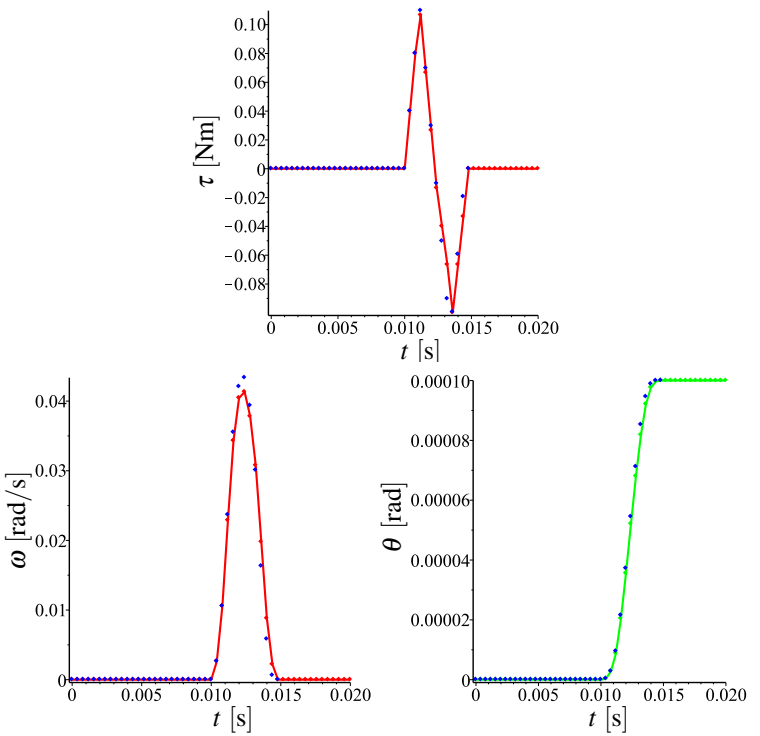

Figure 4.27 This is the generated data by the "Lock to first" approach for going from 0 to 0.0001 rad starting at 0.01 s with $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The blue dots show the corresponding data from the Reflexxes library. Both trajectories reach the finish angle after 0.0048 s.

Looking at the data for the 0.0001 rad step it can be seen that the generated

trajectory reaches the finish position at the same sample-time as the corresponding time-optimal solution from the Reflexxes library, see Fig. 4.27. From Fig. 4.28 it can be seen that the generated trajectory also is feasible since none of the states or the control signal go outside of the limits. As mentioned before it will take the deadbeat controller three extra samples to reach the finish angle. This means that the control signal generated by the controller will not be the time optimal (it will be three time-steps longer than the optimal) and should just be seen as a test of whether or not it is possible to follow the trajectory.

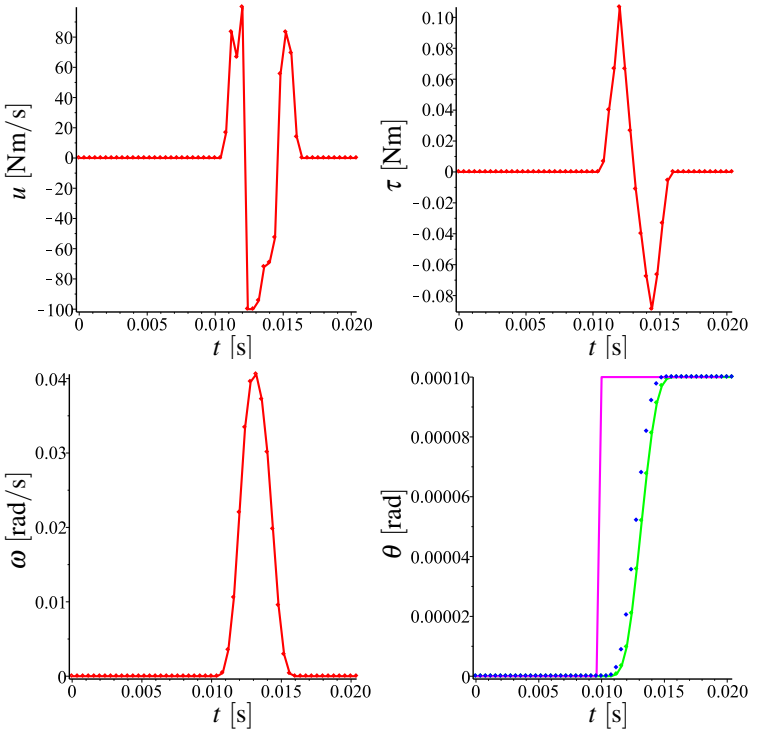

Figure 4.28 This is the simulation results when sending the trajectory from the data in Fig. 4.27, seen as blue dots in the bottom right diagram. The top left diagram shows the control signal generated when sending the trajectory to the deadbeat controller. The system reaches the finish angle after 0.006 s.

For a 0.03 rad step the generated trajectory hits the torque limits but still reaches the finish angle after the same time as the time-optimal, see Fig. 4.29, and as can be seen in Fig. 4.30 this trajectory is also feasible. Figures 4.31 and 4.32 shows the same for a 0.05 rad step when both the velocity and torque hits the limits and also in this case the generated trajectory is time-optimal and feasible.

To test if the "Lock to first" approach broke down and stopped working for larger steps with higher limits a trajectory from 0 to 1 rad with $u_{max} = -u_{min} = 2000$ Nm/s, $\tau_{max} = -\tau_{min} = 10$ Nm and $\omega_{max} = -\omega_{min} = 30$ rad/s was also generated and sent to the deadbeat controller. As can be seen in Fig. 4.33 the approach also works for these parameters.

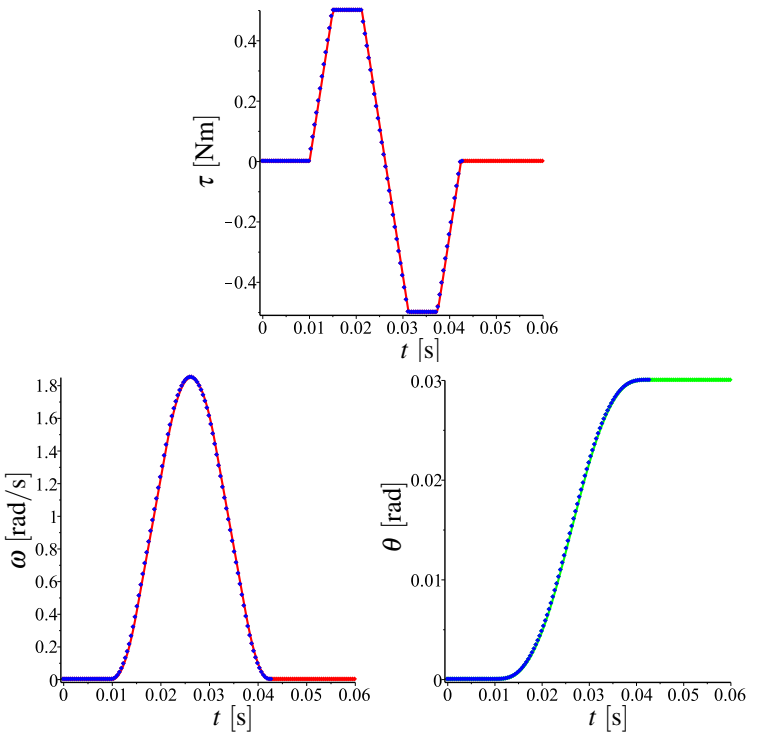

Figure 4.29 This is the generated data by the "Lock to first" approach for going from 0 to 0.03 rad starting at 0.01 s with $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The blue dots show the corresponding data from the Reflexxes library. Both trajectories reach the finish angle after 0.0328 s.

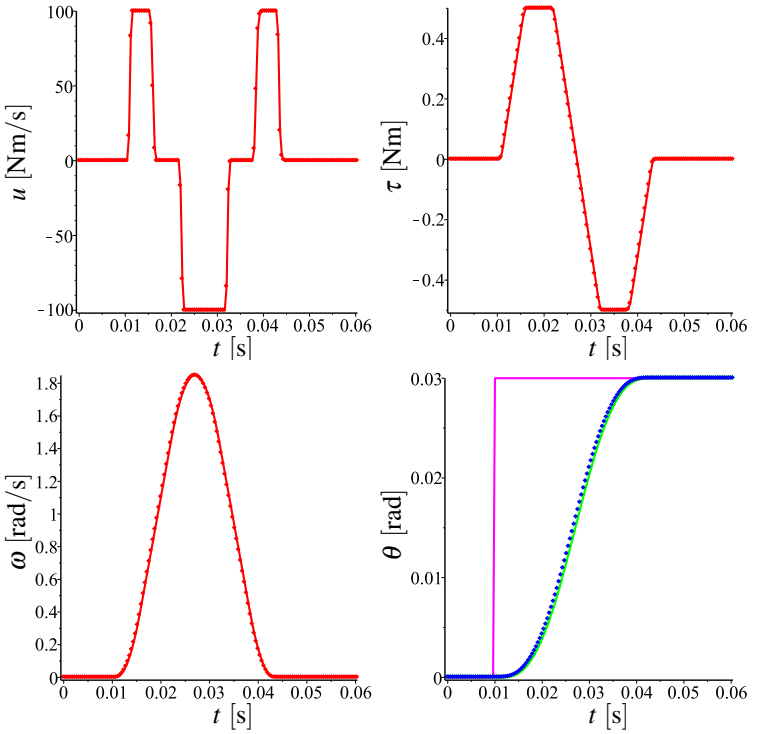

Figure 4.30 This is the simulation results when sending the trajectory from the data in Fig. 4.29, seen as blue dots in the bottom right diagram. The top left diagram shows the control signal generated when sending the trajectory to the deadbeat controller. The system reaches the finish angle after 0.0340 s.

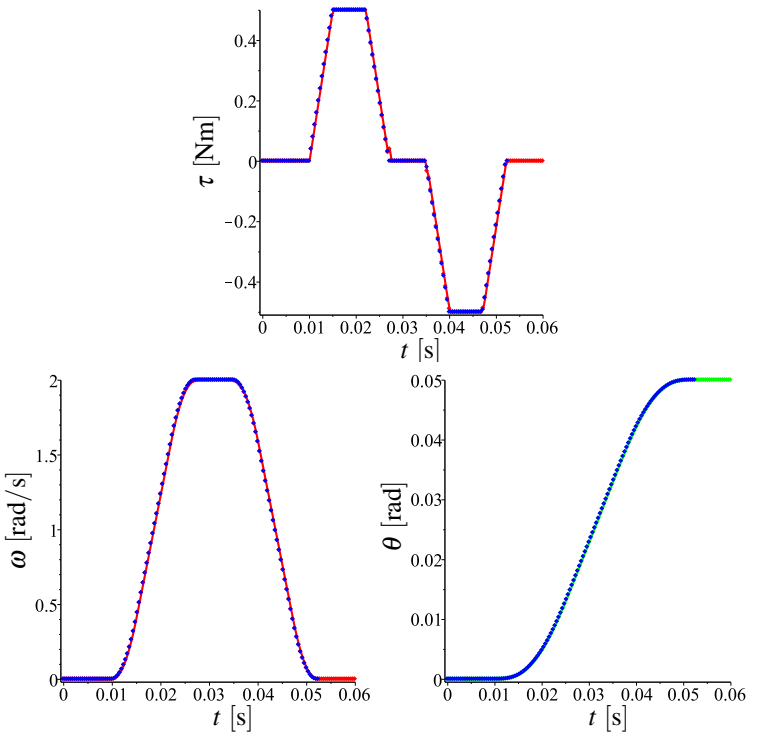

Figure 4.31 This is the generated data by the "Lock to first" approach for going from 0 to 0.05 rad starting at 0.01 s with $u_{max} = -u_{min} = 100$ Nm/s, $\tau_{max} = -\tau_{min} = 0.5$ Nm, $\omega_{max} = -\omega_{min} = 2$ rad/s and $h = 0.4$ ms. The blue dots show the corresponding data from the Reflexxes library. Both trajectories reach the finish angle after 0.0424 s.

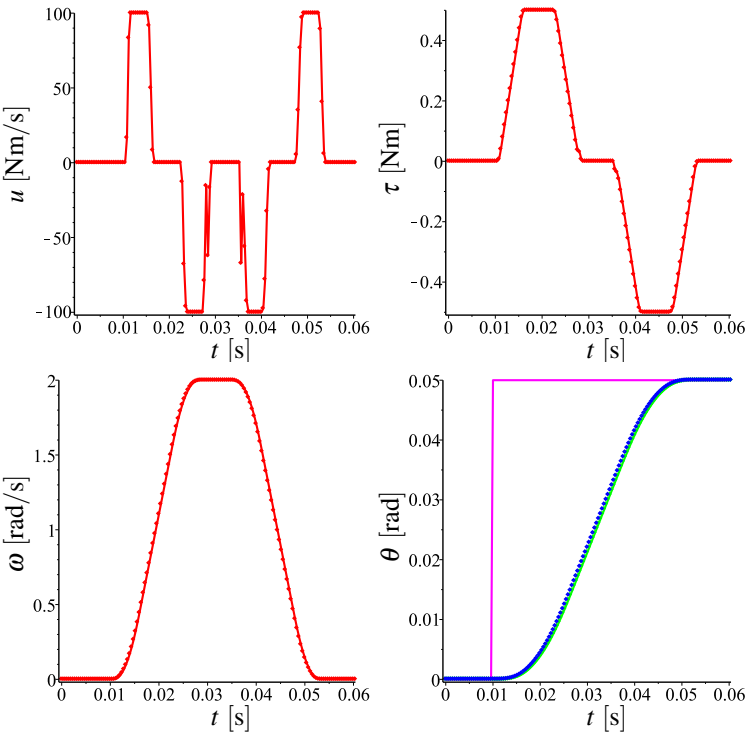

Figure 4.32 This is the simulation results when sending the trajectory from the data in Fig. 4.31, seen as blue dots in the bottom right diagram. The top left diagram shows the control signal generated when sending the trajectory to the deadbeat controller. The system reaches the finish angle after 0.0436 s.

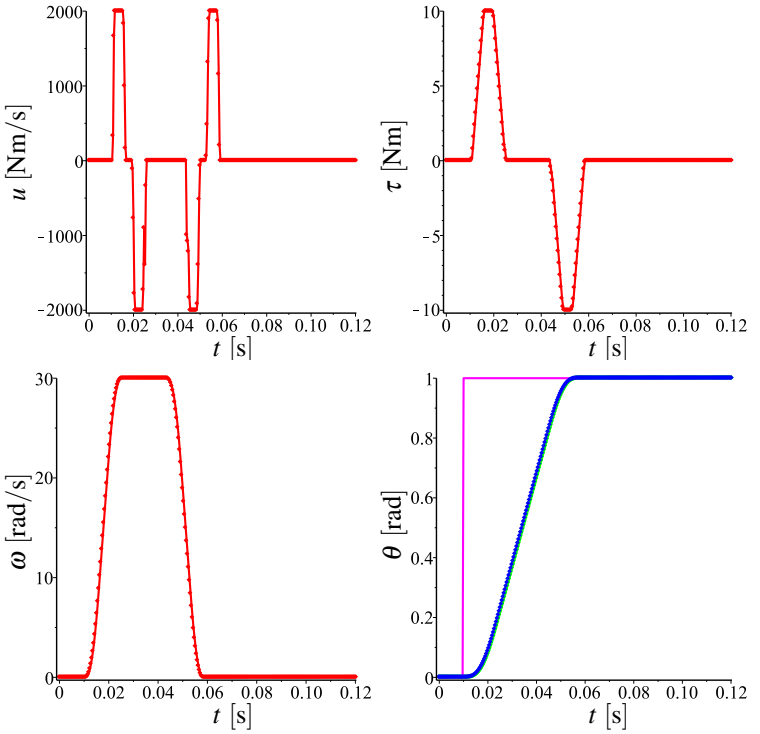

Figure 4.33 This is the simulation results for a "Lock to first" approach generated trajectory going from 0 to 1 rad at 0.01 s, seen as blue dots in the bottom right diagram. The limitation used were $u_{max} = -u_{min} = 2000$ Nm/s, $\tau_{max} = -\tau_{min} = 10$ Nm, $\omega_{max} = -\omega_{min} = 30$ rad/s and $h = 0.4$ ms. The top left diagram shows the control signal generated when sending the trajectory to the deadbeat controller.

5

Conclusions

The approaches “Temporal interpolation”, “State feedback controlled intersection” and “Intersection by control signal” from Secs. 3.4, 3.5 and 3.6 are all based on recalculating the intersection data at each time-step. This generated strange behaviours and poor performance due to closed-loop dynamics. The recalculating also made the approaches unnecessary computationally heavy. Because of this it proved simpler and better to use the “Lock to first” approach which saves the data from the first found intersection and samples this instead of recalculating. With this approach it was possible to generate trajectories (for one axis with jerk, torque and velocity limits) that were as time optimal as the trajectories generated by the Reflexxes library. All the generated trajectories were also feasible in the sense that they were possible to follow by a deadbeat controller, which is the most aggressive discrete-time controller, without reaching the limits. Because of this it was concluded that the “Lock to first” approach is able to generate feasible time-optimal trajectories for one axis with constant inertia and constant limits. However, since this approach, in contrast to the Reflexxes library, does not depend on constant dynamics it can be used to generate time-optimal trajectories for systems with complex dynamics that varies. These variations could for example be that the external torque, on the axis, depends on the current position. With this approach the external torque could be updated in the model along the way when generating the trajectory. Thereby removing the need to complicate the time-optimal solution with the symbolic expression of how the external torque changes. This also opens up the possibility of adding sampled dynamics into the time-optimal solution.

By using the approach on multiple axes and synchronizing them at each time-step, as described in the Sec. 1.2, it could also be possible to get this method to handle multiple axes. If this works then this method of using forward and backward iterating in time could be a fast and cost effective way to generate time-optimal trajectories for robots with non-linear dynamics.

6

Future work

The proposed method of handling multiple axes, described in Sec. 1.2, should be tested. It should also be tested how well the approach actually works with changing dynamics and evaluate what the limits are in terms of, for example, how rapid the change may be. The “Lock to first” approach should be tested for a model without jerk restriction, to make it easier to test with multiple axes and changing dynamics, and the computational power needed by the approach should be evaluated.

The kinematics and the dynamic model of the robot should be tested. It should also be tested to see if the dynamic model could be modified to generate an equivalent moment of inertia and external torque load, instead of just torque. This could then be used in the model for the “Lock to first” approach for multiple axes to see if this approach can be used on the real Flexpicker.

7

Appendix

7.1 Maple code

Forward iterating function

```
#This function iterates one time-step forward.
#stat is the current state vector, jerkInput the control
#signal, inert is the load inertia and
#gTau is the load torque

jerkToPosForwardStep := proc(stat, jerkInput, inert, gTau)
 local output,A,B,jerk, tauRef,
 inputp, wOver, temp;
 A:=Matrix(subs([Inertia=inert],discCurrentSys:-a));
 B:=Matrix(subs([Inertia=inert],discCurrentSys:-b));

 jerk := jerkInput;

 tauRef := 0;

 wOver := ((abs(stat[1]+gTau)/jerkMax+1.0*dt)^2)/2*jerkMax;
 wOver := wOver*33/inert;
 if (abs(stat[3]) +  wOver > wMax) then
  tauRef := -gTau;

  if (stat[1] < tauRef) then
   jerk := jerkMax;

   if (stat[1] + dt * jerk > tauRef) then
    jerk := (tauRef - stat[1]) / dt;
   end if;
```

```
  elif (stat[1] > tauRef) then
   jerk := -jerkMax;

   if (stat[1] + dt * jerk < tauRef) then
    jerk := (tauRef - stat[1]) / dt;
   end if;
  else
   jerk := 0;
  end if;
 end if;

 if (stat[1] + dt * jerk > tauMax) then
  jerk := (tauMax - stat[1]) / dt;
 elif (stat[1] + dt * jerk < -tauMax) then
  jerk := (-tauMax - stat[1]) / dt;
 end if;

 if (jerk > jerkMax) then
  jerk := jerkMax;
 elif (jerk < -jerkMax) then
  jerk := -jerkMax;
 end if;

 output := simplify(A . stat + B . <jerk,gTau>);

 return output;
end proc:
```

Backward iterating function

```
#This function iterates one time-step backward.
#stat is the current state vector, jerkInput the control
#signal, inert is the load inertia and
#gTau is the load torque

jerkToPosBackStep := proc(stat, jerkInput, inert, gTau)
 local output,A,B,jerk,
 tauRef, wOver;
 A:=Matrix(subs([Inertia=inert],backDiscCurrentSys:-a));
 B:=Matrix(subs([Inertia=inert],backDiscCurrentSys:-b));

 jerk := jerkInput;
```

```
 wOver := ((abs(stat[1]+gTau)/jerkMax+1.0*dt)^2)/2*jerkMax;
 wOver := wOver*33/inert;
 if (abs(stat[3]) +  wOver > wMax) then
  tauRef := -gTau;

  if (stat[1] < tauRef) then
   jerk := -jerkMax;

   if (stat[1] - dt * jerk > tauRef) then
    jerk := -(tauRef - stat[1]) / dt;
   end if;
  elif (stat[1] > tauRef) then
   jerk := jerkMax;

   if (stat[1] - dt * jerk < tauRef) then
    jerk := -(tauRef - stat[1]) / dt;
   end if;
  else
   jerk := 0;
  end if;
 end if;

 if (stat[1] - dt * jerk > tauMax) then
  jerk := -(tauMax - stat[1]) / dt;
 elif (stat[1] - dt * jerk < -tauMax) then
  jerk := -(-tauMax - stat[1]) / dt;
 end if;

 if (jerk > jerkMax) then
  jerk := jerkMax;
 elif (jerk < -jerkMax) then
  jerk := -jerkMax;
 end if;

 output := simplify(A . stat + B . <jerk,gTau>);

 return output, jerk;
end proc:
```

Aligning velocity and torque function

```
#This function returns the position when the backwards
#break path has the same velocity and torque as the
```

```
#current state vector.
#currentState is the state to match the velocity and
#torque with, finnishState is the initial state vector for
#the backwards iteration, inert is the load inertia and
#gTau is the load torque.

breakingDataFromTauAndW := proc(currentState, finnishState,
                                          inert, gravTau)
 local output, i, backState, syncTauStat, jerk, inter;

 backState := finnishState;
 syncTauStat := finnishState;
 output := syncTauStat;


 if (currentState[2] < backState[2]) then
  jerk := jerkMax;
 else
  jerk := -jerkMax;
 end if;

 for i from 1 to 200 do

  syncTauStat := syncTau(currentState, backState,
                         -jerk, inert, gravTau);

  if (backState[3] > wMax) or
     (backState[3] < - wMax) then
   break;
  end if;

  if (jerk < 0) then
   if (currentState[3] > syncTauStat[3]) then
    break;
   end if;
  else
   if (currentState[3] < syncTauStat[3]) then
    break;
   end if;
  end if;

  output := syncTauStat;
```

```
  backState := jerkToPosBackStep(backState, jerk,
                      inert, gravTau);
 end do;

 if (i = 200) then
  return [0, output];
 end if;

 if (syncTauStat[3] - output[3] = 0) then
  inter := 0;
 else
  inter := (currentState[3] - output[3]) /
                   (syncTauStat[3] - output[3]);
 end if;

 output := output + (syncTauStat - output) * inter;

 return [1, output];
end proc:
```

7.2 Min movement in positive time

Consider the control signal $u_{pmin}(t)$ which is equal to u_{min} as long as the torque and velocity limits are inactive. According to Eq. (2.36), the following holds before the torque and velocity limits need to be considered.

$$\begin{array}{rcl} u(t) & \geq & u_{pmin}(t) = u_{min}, \quad t_0 \leq t \leq t_1 \\ & \Downarrow & \\ \tau(t) & \geq & \tau_{pmin}(t) = \int_{t_0}^{t} u_{min} ds + \tau(t_0) \\ & \Downarrow & \\ \omega(t) & \geq & \omega_{pmin}(t) = I_{inertia} \int_{t_0}^{t} \tau_{pmin}(s) + \tau_{grav} ds + \omega(t_0) \\ & \Downarrow & \\ \theta(t) & \geq & \theta_{pmin}(t) = \int_{t_0}^{t} \omega_{pmin}(s) ds + \theta(t_0) \end{array} \tag{7.1}$$

where the time t_1 satisfies either

$$\tau_{pmin}(t_1) = \tau_{min} \tag{7.2}$$

or

$$\begin{array}{l} \omega_{pmin}(t_1 + t_b) = \omega_{min} \\ \tau_{pmin}(t_1 + t_b) = -\tau_{grav} \end{array} \tag{7.3}$$

where t_b is the time it takes to increase $\tau_{pmin}(t_1)$ to get $\dot{\omega} = 0$.

If the torque limit is reached first, see Eq. (7.2), then together with Eq. (7.1) the following is true

$$\begin{array}{rcl} & & u_{pmin}(t) = 0, \quad t_1 < t \leq t_2 \\ \tau(t) & \geq & \tau_{pmin}(t) = \tau_{min} \\ & \Downarrow & \\ \omega(t) & \geq & \omega_{pmin}(t) = I_{inertia} \int_{t_1}^{t} \tau_{pmin}(s) + \tau_{grav} ds + \omega_{pmin}(t_1) \\ & \Downarrow & \\ \theta(t) & \geq & \theta_{pmin}(t) = \int_{t_1}^{t} \omega_{pmin}(s) ds + \theta_{pmin}(t_1) \end{array} \tag{7.4}$$

where the time t_2 satisfies

$$\begin{array}{c} \omega_{pmin}(t_2 + t_b) = \omega_{min} \\ \tau_{pmin}(t_2 + t_b) = -\tau_{grav} \end{array} \tag{7.5}$$

If the velocity limit would be reached unless the torque is lowered by applying u_{max}, either after the torque limit has been reached or before

$$\begin{array}{c} t_3 = t_1 \text{ or } t_2 \\ \omega_{pmin}(t_3 + t_b) = \omega_{min} \\ \tau_{pmin}(t_3 + t_b) = -\tau_{grav} \end{array} \tag{7.6}$$

Together with Eqs. (7.1) and (7.4) it is then given that

$$\begin{array}{rcl} \tau(t_3) & \geq & \tau_{pmin}(t_3) \\ \omega(t_3) & \geq & \omega_{pmin}(t_3) \\ \theta(t_3) & \geq & \theta_{pmin}(t_3) \end{array} \tag{7.7}$$

$$\begin{array}{l} u_{pmin}(t) = u_{max}, \quad t_3 < t \leq t_3 + t_b \\ \tau_{pmin}(t) = \int_{t_3}^{t} u_{pmin}(t) ds + \tau_{pmin}(t_3) \\ \omega_{pmin}(t) = I_{inertia} \int_{t_3}^{t} \tau_{pmin}(s) + \tau_{grav} ds + \omega_{pmin}(t_3) \\ \theta_{pmin}(t) = \int_{t_3}^{t} \omega_{pmin}(s) ds + \theta_{pmin}(t_3) \end{array} \tag{7.8}$$

Putting together Eqs. (7.6) and (7.15) gives

$$\omega_{pmin}(t_3 + t_b) = I_{inertia} \int_{t_3}^{t_3 + t_b} \tau_{pmin}(s) + \tau_{grav} ds + \omega_{pmin}(t_3) = \omega_{min} \tag{7.9}$$

$$\begin{array}{rcl} \tau_{pmin}(t_3 + t_b) & = & \int_{t_3}^{t_3 + t_b} u_{max} ds + \tau_{pmin}(t_3) = -\tau_{grav} \\ & \Downarrow & \\ \int_{t_3}^{t_3 + t_b} u_{max} ds & = & u_{max} t_b = -\tau_{pmin}(t_3) - \tau_{grav} \\ & \Downarrow & \\ t_b & = & -\dfrac{\tau_{pmin}(t_3) + \tau_{grav}}{u_{max}} = \end{array} \tag{7.10}$$

Now let us assume that

$$\omega(t) < \omega_{pmin}(t), \;\; t_3 < t \leq t_3 + t_b \tag{7.11}$$

Since $\omega(t_3) \geq \omega_{pmin}(t_3)$, see Eq. (7.7), and $\omega(t) \in C^1, \; \tau(t) \in C^0 \; \forall \; t$ this leads to that there exists a t_i such that

$$\begin{array}{c} \omega(t_i) = \omega_{pmin}(t_i) \\ \tau(t_i) < \tau_{pmin}(t_i) \end{array} \tag{7.12}$$

For $t_i \leq t \leq t_3 + t_b$ then, the following holds

$$\begin{array}{rcl} \tau(t) & = & \int_{t_i}^{t} u(s)ds + \tau(t_i) \leq \\ & \leq & \int_{t_i}^{t} u_{max}ds + \tau(t_i) < \\ & < & \int_{t_i}^{t} u_{max}ds + \tau_{pmin}(t_i) = \tau_{pmin}(t) \end{array} \tag{7.13}$$

this lead to

$$\begin{array}{rcl} \omega(t_3 + t_b) & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega(t_i) = \\ & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega_{pmin}(t_i) < \\ & < & \int_{t_i}^{t_3+t_b} \tau_{pmin}(s)ds + \omega_{pmin}(t_i) = \\ & = & \omega_{pmin}(t_3 + t_b) = \omega_{min} \end{array} \tag{7.14}$$

so $\omega(t) < \omega_{pmin}(t), \;\; t_3 < t \leq t_3 + t_b$ cannot be true without $\omega(t_3 + t_b) < \omega_{min}$ and since this result violates the limits, $\omega(t)$ cannot be less than $\omega_{pmin}(t)$. This leads to

$$\begin{array}{c} u_{pmin}(t) = u_{max}, \;\; t_3 < t \leq t_3 + t_b, \;\; t_b = -\dfrac{\tau_{pmin}(t_3) + \tau_{grav}}{u_{max}} \\ \begin{array}{rcl} \omega(t) & \geq & \omega_{pmin}(t) \\ & \Downarrow & \\ \theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \geq & \theta_{pmin}(t) = \int_{t_3}^{t} \omega_{pmin}(s)ds + \theta_{pmin}(t_3) \end{array} \end{array} \tag{7.15}$$

For $t > t_3 + t_b$

$$\begin{array}{rcl} \omega(t) & \geq & \omega_{pmin}(t) = \omega_{min} \\ & \Downarrow & \\ \theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \geq & \theta_{pmin}(t) = \int_{t_3}^{t} \omega_{pmin}(s)ds + \theta_{pmin}(t_3) \end{array} \tag{7.16}$$

Combining Eqs. (7.1), (7.4), (7.15) and (7.16) the following inequalities are yielded

$$\begin{array}{rcl} \omega(t) & \geq & \omega_{pmin}(t) \\ \theta(t) & \geq & \theta_{pmin}(t) \end{array} \quad \text{for} \;\; t \geq t_0 \tag{7.17}$$

7.3 Max movement in negative time

Consider the control signal $u_{nmax}(t)$ which is equal to u_{min} as long as the torque and velocity limits are inactive. For negative integral times Eq. (2.36) is changed to

$$\begin{array}{c} f_0(s) \leq f_1(s) \leq f_2(s) \; \forall \, t < s < 0 \\ \Downarrow \\ \int_0^t f_2(s)ds \leq \int_0^t f_1(s)ds \leq \int_0^t f_0(s)ds \end{array} \tag{7.18}$$

Then the following holds before the torque and velocity limits need to be considered

$$\begin{array}{rcl} u(t) & \geq & u_{nmax}(t) = u_{min}, \quad t_1 \leq t \leq t_0 \\ & \Downarrow & \\ \tau(t) & \leq & \tau_{nmax}(t) = \int_{t_0}^t u_{min}ds + \tau(t_0) \\ & \Downarrow & \\ \omega(t) & \geq & \omega_{nmax}(t) = I_{inertia} \int_{t_0}^t \tau_{nmax}(s) + \tau_{grav}ds + \omega(t_0) \\ & \Downarrow & \\ \theta(t) & \leq & \theta_{nmax}(t) = \int_{t_0}^t \omega_{nmax}(s)ds + \theta(t_0) \end{array} \tag{7.19}$$

where the time t_1 satisfies either

$$\tau_{nmax}(t_1) = \tau_{max} \tag{7.20}$$

or

$$\begin{array}{c} \omega_{nmax}(t_1 + t_b) = \omega_{min} \\ \tau_{nmax}(t_1 + t_b) = -\tau_{grav} \end{array} \tag{7.21}$$

where t_b is the time it takes to lower $\tau_{nmax}(t_1)$ to reach $\dot{\omega} = 0$.

If the torque limit is reached first, see Eq. (7.20), then together with Eq. (7.19) the following is true

$$\begin{array}{rcl} & & u_{nmax}(t) = 0, \quad t_2 \leq t < t_1 \\ \tau(t) & \leq & \tau_{nmax}(t) = \tau_{max} \\ & \Downarrow & \\ \omega(t) & \geq & \omega_{nmax}(t) = I_{inertia} \int_{t_1}^t \tau_{nmax}(s) + \tau_{grav}ds + \omega_{nmax}(t_1) \\ & \Downarrow & \\ \theta(t) & \leq & \theta_{nmax}(t) = \int_{t_1}^t \omega_{nmax}(s)ds + \theta_{nmax}(t_1) \end{array} \tag{7.22}$$

where the time t_2 satisfies

$$\begin{array}{c} \omega_{nmax}(t_2 + t_b) = \omega_{min} \\ \tau_{nmax}(t_2 + t_b) = -\tau_{grav} \end{array} \tag{7.23}$$

If the velocity limit would be reached unless the torque is lowered by applying u_{max}, either after the torque limit has been reached or before

$$\begin{array}{c} t_3 = t_1 \ \text{ or } \ t_2 \\ \omega_{nmax}(t_3 + t_b) = \omega_{min} \\ \tau_{nmax}(t_3 + t_b) = -\tau_{grav} \end{array} \tag{7.24}$$

Together with Eqs. (7.19) and (7.22) it is then given that

$$\begin{array}{rcl} \tau(t_3) & \leq & \tau_{nmax}(t_3) \\ \omega(t_3) & \geq & \omega_{nmax}(t_3) \\ \theta(t_3) & \leq & \theta_{nmax}(t_3) \end{array} \tag{7.25}$$

$$\begin{array}{rcl} u_{nmax}(t) & = & u_{max}, \quad t_3 + t_b \leq t < t_3 \\ \tau_{nmax}(t) & = & \int_{t_3}^{t} u_{nmax}(t)ds + \tau_{nmax}(t_3) \\ \omega_{nmax}(t) & = & I_{inertia} \int_{t_3}^{t} \tau_{nmax}(s) + \tau_{grav} ds + \omega_{nmax}(t_3) \\ \theta_{nmax}(t) & = & \int_{t_3}^{t} \omega_{nmax}(s)ds + \theta_{nmax}(t_3) \end{array} \tag{7.26}$$

Putting together Eqs. (7.24) and (7.33) gives

$$\omega_{nmax}(t_3 + t_b) = I_{inertia} \int_{t_3}^{t_3+t_b} \tau_{nmax}(s) + \tau_{grav} ds + \omega_{nmax}(t_3) = \omega_{min} \tag{7.27}$$

$$\begin{array}{c} \tau_{nmax}(t_3 + t_b) = \int_{t_3}^{t_3+t_b} u_{max} ds + \tau_{nmax}(t_3) = -\tau_{grav} \\ \Downarrow \\ \int_{t_3}^{t_3+t_b} u_{max} ds = u_{max} t_b = -\tau_{nmax}(t_3) - \tau_{grav} \\ \Downarrow \\ t_b = -\dfrac{\tau_{nmax}(t_3) + \tau_{grav}}{u_{max}} = \end{array} \tag{7.28}$$

Now let us assume that

$$\omega(t) < \omega_{nmax}(t), \text{ for some } \ t_3 + t_b \leq t < t_3 \tag{7.29}$$

Since $\omega(t_3) \geq \omega_{nmax}(t_3)$, see Eq. (7.25), and $\omega(t) \in C^1, \ \tau(t) \in C^0 \ \forall \, t$ this leads to that there exists a t_i such that

$$\begin{array}{c} \omega(t_i) = \omega_{nmax}(t_i) \\ \tau(t_i) > \tau_{nmax}(t_i) \end{array} \tag{7.30}$$

For $t_3 + t_b \leq t \leq t_i$ then, the following holds

$$\begin{array}{rcl} \tau(t) & = & \int_{t_i}^{t} u(s)ds + \tau(t_i) \geq \\ & \geq & \int_{t_i}^{t} u_{max} ds + \tau(t_i) > \\ & > & \int_{t_i}^{t} u_{max} ds + \tau_{nmax}(t_i) = \tau_{nmax}(t) \end{array} \tag{7.31}$$

this lead to

$$\begin{array}{rcl} \omega(t_3+t_b) & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega(t_i) = \\ & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega_{nmax}(t_i) < \\ & < & \int_{t_i}^{t_3+t_b} \tau_{nmax}(s)ds + \omega_{nmax}(t_i) = \\ & = & \omega_{nmax}(t_3+t_b) = \omega_{min} \end{array} \tag{7.32}$$

so $\omega(t) < \omega_{nmax}(t),\ \ t_3+t_b \leq t < t_3$ cannot be true without $\omega(t_3+t_b) < \omega_{min}$ and since this violates the state limits $\omega(t)$ cannot be less than $\omega_{nmax}(t)$. This leads to

$$\begin{array}{rcl} u_{nmax}(t) = u_{max},\ \ t_3+t_b \leq t < t_3,\ \ t_b = -\dfrac{\tau_{nmax}(t_3)+\tau_{grav}}{u_{max}} & & \\ \omega(t) & \geq & \omega_{nmax}(t) \\ & \Downarrow & \\ \theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \leq & \theta_{nmax}(t) = \int_{t_3}^{t} \omega_{nmax}(s)ds + \theta_{nmax}(t_3) \end{array} \tag{7.33}$$

For $t < t_3 + t_b$

$$\begin{array}{rcl} \omega(t) & \geq & \omega_{nmax}(t) = \omega_{min} \\ & \Downarrow & \\ \theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \leq & \theta_{nmax}(t) = \int_{t_3}^{t} \omega_{nmax}(s)ds + \theta_{nmax}(t_3) \end{array} \tag{7.34}$$

Combining Eqs. (7.19), (7.22), (7.33) and (7.34) the following inequalities are yielded

$$\begin{array}{rcl} \omega(t) & \geq & \omega_{nmax}(t) \\ \theta(t) & \leq & \theta_{nmax}(t) \end{array} \quad \text{for } t \leq t_0 \tag{7.35}$$

7.4 Min movement in negative time

Consider the control signal $u_{nmin}(t)$ which is equal to u_{max} as long as the torque and velocity limits are inactive. For negative integral times Eq. (2.36) is changed to

$$\begin{array}{c} f_0(s) \leq f_1(s) \leq f_2(s)\ \forall\, t < s < 0 \\ \Downarrow \\ \int_0^t f_2(s)ds \leq \int_0^t f_1(s)ds \leq \int_0^t f_0(s)ds \end{array} \tag{7.36}$$

Then the following holds before the torque and velocity limits need to be considered

$$\begin{array}{rcl} u(t) & \leq & u_{nmin}(t) = u_{max}, \quad t_1 \leq t \leq t_0 \\ & \Downarrow & \\ \tau(t) & \geq & \tau_{nmin}(t) = \int_{t_0}^{t} u_{max} ds + \tau_{nmin}(t_0) \\ & \Downarrow & \\ \omega(t) & \leq & \omega_{nmin}(t) = I_{inertia} \int_{t_0}^{t} \tau_{nmin}(s) + \tau_{grav} ds + \omega_{nmin}(t_0) \\ & \Downarrow & \\ \theta(t) & \geq & \theta_{nmin}(t) = \int_{t_0}^{t} \omega_{nmin}(s) ds + \theta_{nmin}(t_0) \end{array} \tag{7.37}$$

where the time t_1 satisfies either

$$\tau_{nmin}(t_1) = \tau_{min} \tag{7.38}$$

or

$$\begin{array}{c} \omega_{nmin}(t_1 + t_b) = \omega_{max} \\ \tau_{nmin}(t_1 + t_b) = -\tau_{grav} \end{array} \tag{7.39}$$

where t_b is the time it takes to increase $\tau(t_1)$ to get $\dot{\omega} = 0$.

If the torque limit is reached first, see Eq. (7.38), then together with Eq. (7.37) the following is true

$$\begin{array}{rcl} & & u_{nmin}(t) = 0, \quad t_2 \leq t < t_1 \\ \tau(t) & \geq & \tau_{nmin}(t) = \tau_{min} \\ & \Downarrow & \\ \omega(t) & \leq & \omega_{nmin}(t) = I_{inertia} \int_{t_1}^{t} \tau_{nmin}(s) + \tau_{grav} ds + \omega_{nmin}(t_1) \\ & \Downarrow & \\ \theta(t) & \geq & \theta_{nmin}(t) = \int_{t_1}^{t} \omega_{nmin}(s) ds + \theta_{nmin}(t_1) \end{array} \tag{7.40}$$

where the time t_2 satisfies

$$\begin{array}{c} \omega_{nmin}(t_2 + t_b) = \omega_{max} \\ \tau_{nmin}(t_2 + t_b) = -\tau_{grav} \end{array} \tag{7.41}$$

If the velocity limit would be reached unless the torque is lowered by applying u_{min}, either after the torque limit has been reached or before

$$\begin{array}{c} t_3 = t_1 \text{ or } t_2 \\ \omega_{nmin}(t_3 + t_b) = \omega_{max} \\ \tau_{nmin}(t_3 + t_b) = -\tau_{grav} \end{array} \tag{7.42}$$

Together with Eqs. (7.37) and (7.40) it is then given that

$$\begin{array}{rcl} \tau(t_3) & \geq & \tau_{nmin}(t_3) \\ \omega(t_3) & \leq & \omega_{nmin}(t_3) \\ \theta(t_3) & \geq & \theta_{nmin}(t_3) \end{array} \tag{7.43}$$

$$\begin{array}{rcl} u_{nmin}(t) & = & u_{min}, \quad t_3 + t_b \leq t < t_3 \\ \tau_{nmin}(t) & = & \int_{t_3}^{t} u_{nmin}(t)ds + \tau_{nmin}(t_3) \\ \omega_{nmin}(t) & = & I_{inertia} \int_{t_3}^{t} \tau_{nmin}(s) + \tau_{grav} ds + \omega_{nmin}(t_3) \\ \theta_{nmin}(t) & = & \int_{t_3}^{t} \omega_{nmin}(s)ds + \theta_{nmin}(t_3) \end{array} \tag{7.44}$$

Putting together Eqs. (7.42) and (7.51) gives

$$\omega_{nmin}(t_3 + t_b) = I_{inertia} \int_{t_3}^{t_3+t_b} \tau_{nmin}(s) + \tau_{grav} ds + \omega_{nmin}(t_3) = \omega_{max} \tag{7.45}$$

$$\begin{array}{rcl} \tau_{nmin}(t_3 + t_b) & = & \int_{t_3}^{t_3+t_b} u_{min} ds + \tau_{nmin}(t_3) = -\tau_{grav} \\ & \Downarrow & \\ \int_{t_3}^{t_3+t_b} u_{min} ds & = & u_{min} t_b = -\tau_{nmin}(t_3) - \tau_{grav} \\ & \Downarrow & \\ t_b & = & -\dfrac{\tau_{nmin}(t_3) + \tau_{grav}}{u_{min}} = \end{array} \tag{7.46}$$

Now let us assume that

$$\omega(t) > \omega_{nmin}(t), \text{ for some } t_3 + t_b \leq t < t_3 \tag{7.47}$$

Since $\omega(t_3) \leq \omega_{nmin}(t_3)$, see Eq. (7.43), and $\omega(t) \in C^1,\ \tau(t) \in C^0\ \forall\ t$ this leads to that there exists a t_i such that

$$\begin{array}{c} \omega(t_i) = \omega_{nmin}(t_i) \\ \tau(t_i) < \tau_{nmin}(t_i) \end{array} \tag{7.48}$$

For $t_3 + t_b \leq t \leq t_i$ then, the following holds

$$\begin{array}{rcl} \tau(t) & = & \int_{t_i}^{t} u(s)ds + \tau(t_i) \leq \\ & \leq & \int_{t_i}^{t} u_{min} ds + \tau(t_i) < \\ & < & \int_{t_i}^{t} u_{min} ds + \tau_{nmin}(t_i) = \tau_{nmin}(t) \end{array} \tag{7.49}$$

this lead to

$$\begin{array}{rcl} \omega(t_3 + t_b) & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega(t_i) = \\ & = & \int_{t_i}^{t_3+t_b} \tau(s)ds + \omega_{nmin}(t_i) > \\ & > & \int_{t_i}^{t_3+t_b} \tau_{nmin}(s)ds + \omega_{nmin}(t_i) = \\ & = & \omega_{nmin}(t_3 + t_b) = \omega_{max} \end{array} \tag{7.50}$$

So $\omega(t) > \omega_{nmin}(t),\ \ t_3 + t_b \leq t < t_3$ cannot be true without $\omega(t_3 + t_b) > \omega_{max}$ and since this result violates the limits, $\omega(t)$ cannot be greater than $\omega_{nmin}(t)$. This leads to

$$
\begin{array}{rcl}
u_{nmin}(t) = u_{min},\ \ t_3 + t_b \leq t < t_3,\ \ t_b = -\dfrac{\tau_{nmin}(t_3) + \tau_{grav}}{u_{min}} & & \\
\omega(t) & \leq & \omega_{nmin}(t) \\
& \Downarrow & \\
\theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \geq & \theta_{nmin}(t) = \int_{t_3}^{t} \omega_{nmin}(s)ds + \theta_{nmin}(t_3)
\end{array}
\tag{7.51}
$$

For $t < t_3 + t_b$

$$
\begin{array}{rcl}
\omega(t) & \leq & \omega_{nmin}(t) = \omega_{max} \\
& \Downarrow & \\
\theta(t) = \int_{t_3}^{t} \omega(s)ds + \theta(t_3) & \geq & \theta_{nmin}(t) = \int_{t_3}^{t} \omega_{nmin}(s)ds + \theta_{nmin}(t_3)
\end{array}
\tag{7.52}
$$

Combining Eqs. (7.37), (7.40), (7.51) and (7.52) the following inequalities are yielded

$$
\begin{array}{rcl}
\omega(t) & \leq & \omega_{nmin}(t) \\
\theta(t) & \geq & \theta_{nmin}(t)
\end{array}
\quad \text{for } t \leq t_0
\tag{7.53}
$$

7.5 Sync case for min control signal case

For the $u_{pmin}(t)$ synchronization it can be said

$$
\begin{array}{rcl}
\omega_{sync}(t) & = & I \int_0^t (\int_0^r (u_{max})ds + \tau_{forward} + \tau_{grav})dr + \omega_{forward} \\
& \geq & I \int_0^t (\int_0^r (u_{pmin}(s))ds + \tau_{forward} + \tau_{grav})dr + \omega_{forward} \\
& = & \omega_{forward\ pmin}(t) \geq \omega_{min}
\end{array}
\tag{7.54}
$$

so $\omega_{sync}(t) \geq \omega_{min}$. Since $\ddot{\omega}_{sync}(t) \propto u_{sync}(t) = u_{max} > 0$ for $t_{forward} \leq t \leq t_{forward} + t_{sync}$ it means that $\omega_{sync}(t)$ may only have one extremum and in case it exists it is a minimum. This means that $\omega_{sync}(t)$ will have its maximum at $t = t_{forward}$ or $t = t_{forward} + t_{sync}$ and at this points it should be equal to $\omega_{forward}$ and $\omega_{backward}$. So the $\omega_{sync}(t)$ cannot exceed the velocity limits with feasible initial conditions.

Bibliography

[1] *Acopos user's manual v 1.3.1*. B&R Automation. 2014. URL: http://www.motorsystems.com/docs/ds_acopos1022_1045_1090.pdf.

[2] K. J. Åström and B. Wittenmark. *Computer Controlled Systems, Theory and Design*. Prentice Hall, Engelwood Cliffs, NJ, 1984.

[3] R.E Bellman. *Dynamic Programming*. Princeton University Press, Republished 2003.

[4] *Brushless motor basics*. Copley Controls Corp. 2014. URL: http://www.copleycontrols.com/motion/pdf/Brushless_Motor_Basics.pdf.

[5] A.E. Bryson and Ho Yu-Chi. *Applied Optimal Control: Optimization, Estimation and Control*. Halsted Press book. Taylor & Francis, 1975. ISBN: 9780891162285.

[6] Ola Dahl. *Path Constrained Robot Control*. PhD thesis ISRN LUTFD2/TFRT--1038--SE. Department of Automatic Control, Lund University, Sweden, Apr. 1992.

[7] R. Johansson. *System Modeling and Identification*. Prentice Hall, Englewood Cliffs, NJ, 2012.

[8] *Maplesim: technological superiority in multi-domain physical modeling and simulation*. Maplesoft. 2014. URL: http://www.maplesoft.com/view.aspx?SF=7032/195552/msim_whitepaper.pdf.

[9] A. Persson and L. C. Böiers. *Analys i En Variabel*. Studentlitteratur AB, Lund, 2001.

[10] *Product specification IRB 340*. ABB Robotics Products AB. 2014. URL: http://www05.abb.com/global/scot/scot352.nsf/veritydisplay/a51536e64dbeff85c12576cb00528f01/$file/Product%20specification%20340%20M98%20BWOS3.2.pdf.

[11] *Reflexxes motion libraries manual and documentation*. Reflexxes GmbH. 2014. URL: http://www.reflexxes.com/papers/ReflexxesICRA2011.pdf.

[12] *Reflexxes motion library iv rml 1.3.2 academic version*. Reflexxes GmbH. 2014. URL: http://www.reflexxes.com.

[13] K. Rosquist. *Modelling and Control of a Parallel Kinematic Robot*. Master's Thesis TFRT--5929. Department of Automatic Control, Lund University, Sweden, 2013.

[14] *X20cp148x and x20cp348x*. B&R Automation. 2014. URL: http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000232421/X20CPx48x-ENG.pdf.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER´S THESIS

Date of issue
June 2014

Document Number
ISRN LUTFD2/TFRT--5944--SE

Author(s)
Adam Bäckström

Supervisor
Peter Valdt, B&R Automation
Klas Nilsson, Dept. of Computer Science, Lund University, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle
Time-Optimal Control by Iterating Forward and Backward in Time

Abstract

When costumers look for a robot for their factory two things are important. The robot should be as efficient as possible, but still be cheap. In order to make the robot efficient the robot-controller has to know the dynamics of the robot and its limits. Based on these it can then generate a time-optimal plan (trajectory) for each movement. The standard way of generating a time-optimal trajectory with the exact dynamics and limits is very computationally heavy. Today most approaches do the planning based on the hardest limits on each axis of the robot. These values are then used even though the current limits might allow the robot to move faster. This means that the full capacity of the robot will not be utilized and because of this the efficiency is lowered.

In this thesis a different method for generating time-optimal trajectories is tested. The approach is based on iterating forward and backward in time and finding the point where the two paths meet. This approach has the advantage of that it is based on simulating the system. Therefore more complex dynamics can be included in the planning by just calculating a value instead of complicating the optimization problem. Another benefit is that robot manufacturers usually create simulation models of new robots already. This means that very little extra effort would be needed to create the trajectory generator using this approach and this reduces development costs.

Four different approaches for patching together the forward and backward paths are discussed in the thesis. The different techniques are tested on a simplified model of one servo axis of a robot and compared against a known time optimal solution for the simplified model. One of the techniques shows very good results and generates trajectories that are time-optimal.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-99

Recipient's notes

Security classification